NBSIR 83-2639

Cost-Benefit Impact Study on the Adoption of the Draft Proposed Revised X3.23 American National Standard Programming Language COBOL

March 1983

Prepared for
U.S. DEPARTMENT OF COMMERCE
Center for Programming Science and Technology
Data Management and Programming Languages Division
Washington, DC 20234

NBSIR 83-2639

COST-BENEFIT IMPACT STUDY ON THE ADOPTION OF THE DRAFT PROPOSED REVISED X3.23 AMERICAN NATIONAL STANDARD PROGRAMMING LANGUAGE COBOL

Marco Fiorello

Fiorello, Shaw, and Associates

John Cugini

Institute for Computer Science and Technology

March 1983

Prepared for
U.S. DEPARTMENT OF COMMERCE
Center for Programming Science and Technology
Data Management and Programming Languages Division
Washington, DC 20234

U.S. DEPARTMENT OF COMMERCE, Malcolm Baldrige, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

TABLE OF CONTENTS

EXHIBITS:

ABSTRACT

The purpose of the study is to assess the estimated costs and benefits for the Federal Government which would result from adoption of the proposed revision of American National Standard COBOL as a Federal Information Processing Standard (FIPS). Potential benefits of $90.2 million have been identified, stemming primarily from improved productivity in both the development and maintenance of COBOL programs. Estimated costs of $17.9 million have been identified, arising principally from the effort needed to convert old COBOL programs to the new specification, which is incompatible in some respects with the current specification. In support of the study, we conducted interviews with Federal ADP managers and officials, and also analyzed over one thousand Federal COBOL programs for various syntactic characteristics. The study concludes that the potential benefits of a new standard outweigh the estimated costs.

Key words: COBOL; compatibility of programming language standards; conversion costs for COBOL programs; cost-benefit analysis of COBOL standards; Federal use of COBOL; FIPS for COBOL; standardization of COBOL.

EXECUTIVE SUMMARY

The revision of a widely-used language standard, such as COBOL, necessarily involves a trade-off between enhancement of the language and compatibility with existing code.

Potential costs are concentrated in the conversion of old code. This cost depends strongly on the frequency and complexity of the various changes being made to COBOL.

Potential benefits are expected both in the development of new code and the maintenance of existing code, as modern coding techniques become prevalent in the Federal Government.

While not exact, it is possible to make a quantitative estimate of these costs and benefits. The result of the analysis herein is a potential cost of $17.9 million and estimated benefits of $90.2 million (net benefit: $72.3 million).

The maximization of benefits and minimization of costs depends strongly on good DP management practice. In particular, DP staff should be encouraged to take advantage of the new features provided in the revision of COBOL, and there should be a systematic effort to discourage the use of features which are being dropped from the standard.

1.0 INTRODUCTION

1.1 Background And Overview

The goals of the ICST Federal ADP standards program are to develop and issue standards and guidelines for competitive and economic procurement of data processing equipment, efficient management and utilization of ADP, improved protection of ADP resources, and increased productivity of the Federal work force.

This analysis concerns Federal Information Processing Standard (FIPS) COBOL, one of the high level programming language standards. The basic objectives of these standards are: (1) to achieve the long-recognized advantages that are inherent in the use of higher level languages, and (2) to maximize and protect program investments by making it easier and less expensive to exchange programs among different computer systems, including replacement systems.

In developing and maintaining standards and guidelines for ADP activities within the Federal government, ICST works closely with voluntary standards bodies, such as the American National Standards Institute (ANSI) and the International Standards Organization (ISO). FIPS 21-1, the current Federal standard for COBOL, is based on ANSI standard X3.23-1974 [ANS-74] (often called COBOL-74). ANSI committee X3J4 has proposed a revision to COBOL-74, and it is this revision which is under review by various ANSI participants, including ICST. Such a revision is of interest to ICST since it represents a potential basis for a revised FIPS for COBOL.

This study is presented in four major sections and four appendices. In the rest of this section, the study objectives, scope and qualifications, and rationale for the proposed changes are presented. Section two identifies the major technical issues concerning the proposed changes. Section three presents the study methodology and analysis used to determine the cost and benefits. Lastly, section four presents the interpretations of the study findings and the recommendations. The references follow the main body of the report. Appendix A contains an outline of the study methodology, Appendix B contains a list of the substantive changes potentially affecting existing COBOL programs, Appendix C provides a list of the agencies visited and the agencies which supplied sample programs, and Appendix D describes the automated system which was used to analyze the sample programs.

1.2 Study Objectives

The specific objectives of this study are to assess the costs and benefits to the Federal ADP community of the proposed changes to COBOL-74, and to recommend one of the following actions: (a) reject the proposed revision and continue with FIPS

21-1 (this is the status quo, or so-called "Base Case" option), (b) adopt the ANS X3.23 proposed revision as a new FIPS (21-2), or (c) attempt to modify the revision so as to limit costs or enhance benefits to the Federal Government, thus establishing it as a more valuable base for a new FIPS.

1.3 Study Scope And Qualifications

This study scope is limited to the COBOL-related impacts on the Federal ADP community. Of course similar impacts may be expected in the private sector insofar as the characteristics of its COBOL usage resemble those of the Federal sector.

In this analysis, we are concerned with impacts that may result if the proposed changes to ANSI COBOL-74 are also adopted in the FIPS for COBOL. Data available on applications software development and maintenance in the Federal government are general and approximate in nature, and are particularly limited regarding any one specific programming language such as COBOL (although COBOL is by far the most commonly used language within the Federal government, and therefore can hardly be regarded as atypical). We augmented the available general data with staff interviews at nine Federal agencies and with a detailed analysis of a sample of 1068 COBOL programs from eleven Federal agencies. Appendix C shows which agencies participated in the interviews and which contributed sample programs.

The more detailed information obtained from the interviews and sample program analysis enabled us to estimate the magnitudes of the likely costs and benefits attributable to the adoption of the proposed, revised COBOL standard. However, these estimates are not intended as precise determinations of those costs and benefits, as they necessarily reflect many approximations and subjective inputs and interpretations.

The revision of the ANSI COBOL Standard is an ongoing process at the time of this writing, and the conclusions of this report are accordingly subject to change. ANSI committee X3J4 has the responsibility for acting on public comments and a definitive evaluation must wait until the revision takes final shape. If warranted by further changes in the proposed revision, or by additional data gathered from Federal agencies, this report may be updated at a later time to reflect those changes.

The principal assumptions and specifications used in this study include:

1. The new FIPS will be introduced in 1984 and the first compilers will be available in 1984 - 1985.

2. The study horizon is 10 years, 1984 - 1993.

3. Static relationships are appropriate, and the statistics and trends derived from the 1979 - 81 Federal ADP data are representative for the study horizon.

4. A new FIPS for COBOL would be complied with, converted to, and utilized in the same way FIPS 21-1 (COBOL-74) was, relative to FIPS 21 (COBOL-68).

5. The impacts will be based on the changes proposed in [ANS-81], (the blue book) which was current as of September 1981 and upon relevant subsequent actions by ANSC/X3J4.

6. The impact of a revised COBOL standard will be assessed relative to the existing standard FIPS 21-1 (COBOL-74). Thus, as far as possible, any impacts stemming from COBOL-74 itself will be discounted, since its adoption is not at issue.

7. The cost estimates will be made in 1983 constant dollars.

1.4 Reasons For Revising The Current COBOL Standard

The reasons most frequently given in support of the proposed changes to the current ANSI COBOL standard are that they will improve the language's capabilities and ease-of-use, and limit or remove error-prone, useless, and redundant features [NELS-81a], [NELS-81b], [DUBN-81]. The intent is to enhance application programming productivity, facilitate program portability and improve program maintainability.

Notwithstanding the apparent compatibility of the above considerations with the official objectives of FIPS COBOL, the bottom-line criterion by which a new COBOL standard must be judged is whether or not the Federal ADP community will be better off adopting the proposed changes, as compared to the current way of doing business.

2.0 TECHNICAL ISSUES

In this section the proposed changes are identified, and various related technical issues are discussed.

2.1 Substantive Changes And Related Issues

Of interest in this analysis are those proposed changes that are expected to have a substantive effect on existing programs. Also, enhancements to the language (which do not affect existing programs) are examined for their potential impact.

The basic technical issues related to all the proposed changes fall into four categories:

1. Productivity - How do the changes affect COBOL's scope of application and use of new programming techniques?

2. Maintenance - How do the changes affect COBOL's error-prone and ambiguous features and the maintainability of programs?

3. Conversion - How do the changes affect the portability of COBOL programs across various systems? Do they protect the investment in existing COBOL programs?

4. Training - Do the changes make it easier to learn and remember the COBOL language?

Without question, the dominant technical issue on the cost side is that of conversion. This cost manifests itself as the extra effort needed to purge existing programs of code dependent on features of COBOL-74 which have been deleted or changed. As opposed to this cost, which is concentrated and visible, the benefits are necessarily somewhat more speculative and dispersed more evenly throughout Federal ADP practice (and are accordingly more difficult to measure exactly).

3.0 METHODOLOGY AND ANALYSIS

3.1 Framework

The cost-benefit analysis framework used in this study is based on the preliminary guidelines presented in [FIOR-78]:

1. Specification of the Goals and Objectives
2. Preparing the Standard Definition Statement
3. Defining the Base Case
4. Selecting the Cost-Benefit Impact Areas
5. Constructing the Model
6. Obtaining the Data
7. Estimating and Evaluating the Cost-Benefit
8. Presenting and Interpreting the Results

Steps 1 and 2 are discussed in Sections 1 and 2, pertinent results from steps 3-7 are presented below in this Section, and step 8 is discussed in Section 4 on Findings.

3.2 Base Case Statistics

The Base Case statistics are derived from various reference materials, cited at the end of this document, and the study survey and program sample. The pertinent statistics are summarized in Exhibits 3-1 to 3-5, and brief descriptions are given below.

3.2.1 Programmer Pool -

- For the past 10 years the number of Federal agency staff programmers has remained fairly steady in the range of 118,000 to 120,000 staff-years [NBS-81].
- Of those work-years, roughly 60% were primarily for COBOL-related activities in 1980, with a growth to 65% projected for 1985 [GRAY-81].
- Depending upon the Federal agency, the annual programmer turnover rate will vary from a low of 10% to a high of 30%. A reasonable average appears to be 20%.

- All of the agencies contacted have or arrange training courses for new programmer staff.

- In most installations, more than half of the staff are devoted to maintenance (corrective, adaptive and perfective) activities which reflects the life cycle distribution of application software costs [LISW-81], [GAO-81b], [BOEH-81].

- Based on very limited data, it appears that on the average a programmer spends 15% to 25% of available time performing the coding function.

3.2.2 COBOL Program Inventory -

- There are roughly 500,000 application software programs in the Federal inventory. Of these, 50 to 60% are in some form of COBOL.

- Very few, 5-10%, of these 250,000 to 300,000 COBOL programs are in full conformance with the current COBOL FIPS 21-1 [SHOE-80].

- The average COBOL program in our sample contains about 1270 lines of source code (see Exhibit 3-2) and was developed about 6.0 years ago (see Exhibit 3-3). This latter figure compares reasonably well with the 5.4 year estimate in [GAO-81b].

In our sample of 1068 COBOL programs, with over 1.3 million lines of code from 11 Federal agencies, roughly 80% use one or more of the 50 proposed incompatible changes analyzed in this study. If we discount the somewhat special case of the incompatibility concerning the DISPLAY verb (see below), this figure drops to about 40%. The specific frequencies of occurrence are shown in Exhibit 3-4. See Appendix B for a description of the incompatibilities, by number.

Note that in Exhibit 3-4 the statistics are broken down by contributing agency, as well as by individual incompatibility. In cases where there were two separate systems within one agency, these systems were kept separate in the figure; thus, eleven agencies accounted for thirteen systems. Since each agency contributed a different number of programs, there is some question as to whether the summary statistics should be weighted by program, or by agency. Both are computed and displayed. Where necessary for the analysis, we used a reasonable intermediate figure. Of course, the greater the disagreement between the two averages, the greater the uncertainty about what percentage is truly representative of the entire Federal inventory.

Exhibit 3-1 Base Case Parameters

PARAMETER		VALUE ESTIMATES (1982)			SOURCE/BASIS OF ESTIMATE
SYMBOL	DESCRIPTION	PESSIMISTIC	MOST LIKELY (MODE)	OPTIMISTIC	
GC	Government cost to develop FIPS	-	$200,000/yr.	-	ICST Program Best Estimate
NCP	No. Federal COBOL Programs	200,000	250,000	300,000	Study Survey
NLP	No. Lines/Program	1300	1270	1240	Study Sample [DUCF-75]
NDP	No. Federal Staff Programmer-years	115,000	118,000	120,000	[GRAY-81], [NBS-81]
PPM	% of ADP staff time coding	15%	20%	25%	[GRAY-81], [BOEH-81]
CSP	% of ADP staff working on COBOL	45%	50%	60%	[GRAY-81]
PAS	Federal Programmer Annual Salary	(Computer Specialist, GS-11/5, $27,800, unadjusted)			[FCSC-82]
SDY	Available staff-days per staff-year	-	213.2	-	[FCSC-82]
PDT	% of programmer time on new development	50%	40%	30%	[GAO-81b]
PMT	% of programmer time on maintenance	50%	60%	70%	[GAO-81b]

Exhibit 3-2 Distribution of Programs by Size

```
LINES/NO. PGMS/CUMULATIVE % PGMS/CUMULATIVE % LINES

 100/ 36/  3.37%/  0.19% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 200/ 84/ 11.24%/  1.11% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 300/ 94/ 20.04%/  2.82% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 400/ 86/ 28.09%/  5.02% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 500/ 75/ 35.11%/  7.45% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 600/ 74/ 42.04%/ 10.44% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 700/ 68/ 48.41%/ 13.70% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 800/ 57/ 53.75%/ 16.83% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 900/ 49/ 58.33%/ 19.87% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1000/ 43/ 62.36%/ 22.88% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1100/ 38/ 65.92%/ 25.78% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1200/ 21/ 67.88%/ 27.54% XXXXXXXXXXXXXXXXXXXXX
1300/ 27/ 70.41%/ 30.00% XXXXXXXXXXXXXXXXXXXXXXXXXXX
1400/ 28/ 73.03%/ 32.74% XXXXXXXXXXXXXXXXXXXXXXXXXXXX
1500/ 24/ 75.28%/ 35.29% XXXXXXXXXXXXXXXXXXXXXXXX
1600/ 19/ 77.06%/ 37.47% XXXXXXXXXXXXXXXXXXX
1700/ 20/ 78.93%/ 39.86% XXXXXXXXXXXXXXXXXXXX
1800/ 12/ 80.06%/ 41.40% XXXXXXXXXXXX
1900/ 15/ 81.46%/ 43.42% XXXXXXXXXXXXXXX
2000/ 10/ 82.40%/ 44.84% XXXXXXXXXX
2100/  9/ 83.24%/ 46.19% XXXXXXXXX
2200/  6/ 83.80%/ 47.13% XXXXXX
2300/ 20/ 85.67%/ 50.42% XXXXXXXXXXXXXXXXXXXX
2400/ 11/ 86.70%/ 52.30% XXXXXXXXXXX
2500/ 10/ 87.64%/ 54.08% XXXXXXXXXX
2600/  7/ 88.30%/ 55.38% XXXXXXX
2700/  8/ 89.04%/ 56.92% XXXXXXXX
2800/  9/ 89.89%/ 58.73% XXXXXXXXX
2900/  7/ 90.54%/ 60.18% XXXXXXX
3000/  1/ 90.64%/ 60.40% X
3100/  5/ 91.10%/ 61.52% XXXXX
3200/  8/ 91.85%/ 63.36% XXXXXXXX
3300/  3/ 92.13%/ 64.07% XXX
3400/  7/ 92.79%/ 65.79% XXXXXXX
3500/  2/ 92.98%/ 66.30% XX
3600/  4/ 93.35%/ 67.34% XXXX
3700/  2/ 93.54%/ 67.87% XX
3800/  2/ 93.73%/ 68.42% XX
3900/  2/ 93.91%/ 68.98% XX
4000/  2/ 94.10%/ 69.55% XX
4100/  2/ 94.29%/ 70.15% XX
4200/  1/ 94.38%/ 70.45% X
4300/  2/ 94.57%/ 71.07% XX
4400/  5/ 95.04%/ 72.66% XXXXX
4500/  3/ 95.32%/ 73.63% XXX
4600/  2/ 95.51%/ 74.29% XX
4700/  1/ 95.60%/ 74.63% X
4800/  2/ 95.79%/ 75.32% XX
4900/  4/ 96.16%/ 76.75% XXXX
5000/  2/ 96.35%/ 77.46% XX
5100/  3/ 96.63%/ 78.57% XXX
5200/  0/ 96.63%/ 78.57%
5300/  0/ 96.63%/ 78.57%
5400/  3/ 96.91%/ 79.74% XXX
5500/  1/ 97.00%/ 80.14% X
```

Exhibit 3-2 (continued)

```
 5600/  2/ 97.19%/ 80.95% XX
 5700/  2/ 97.38%/ 81.77% XX
 5800/  0/ 97.38%/ 81.77%
 5900/  0/ 97.38%/ 81.77%
 6000/  1/ 97.47%/ 82.20% X
 6100/  1/ 97.57%/ 82.65% X
 6200/  0/ 97.57%/ 82.65%
 6300/  1/ 97.66%/ 83.11% X
 6400/  0/ 97.66%/ 83.11%
 6500/  0/ 97.66%/ 83.11%
 6600/  3/ 97.94%/ 84.54% XXX
 6700/  1/ 98.03%/ 85.03% X
 6800/  0/ 98.03%/ 85.03%
 6900/  0/ 98.03%/ 85.03%
 7000/  1/ 98.13%/ 85.53% X
 7100/  2/ 98.31%/ 86.56% XX
 7200/  1/ 98.41%/ 87.08% X
 7300/  0/ 98.41%/ 87.08%
 7400/  0/ 98.41%/ 87.08%
 7500/  0/ 98.41%/ 87.08%
 7600/  0/ 98.41%/ 87.08%
 7700/  1/ 98.50%/ 87.64% X
 7800/  1/ 98.60%/ 88.21% X
 7900/  0/ 98.60%/ 88.21%
 8000/  1/ 98.69%/ 88.79% X
 8100/  1/ 98.78%/ 89.39% X
 8200/  1/ 98.88%/ 89.98% X
 8300/  0/ 98.88%/ 89.98%
 8400/  1/ 98.97%/ 90.59% X
 8500/  0/ 98.97%/ 90.59%
 8600/  1/ 99.06%/ 91.21% X
 8700/  0/ 99.06%/ 91.21%
 8800/  1/ 99.16%/ 91.85% X
 8900/  0/ 99.16%/ 91.85%
 9000/  2/ 99.34%/ 93.16% XX
 9100/  0/ 99.34%/ 93.16%
 9200/  0/ 99.34%/ 93.16%
 9300/  1/ 99.44%/ 93.83% X
   ...
10800/  1/ 99.53%/ 94.62% X
   ...
11900/  1/ 99.63%/ 95.49% X
   ...
13500/  1/ 99.72%/ 96.47% X
   ...
14200/  1/ 99.81%/ 97.50% X
   ...
17100/  1/ 99.91%/ 98.74% X
   ...
17300/  1/100.00%/100.00% X
```

Exhibit 3-3 Distribution of Programs by Age

MONTHS/NO. PGMS/CUMULATIVE % PGMS

```
  6/ 15/  2.41% XXXXXXXXXXXXXXX
 12/ 37/  8.36% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 18/ 24/ 12.22% XXXXXXXXXXXXXXXXXXXXXXXX
 24/ 44/ 19.29% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 30/ 30/ 24.12% XXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 36/ 36/ 29.90% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 42/ 25/ 33.92% XXXXXXXXXXXXXXXXXXXXXXXXX
 48/ 25/ 37.94% XXXXXXXXXXXXXXXXXXXXXXXXX
 54/ 23/ 41.64% XXXXXXXXXXXXXXXXXXXXXXX
 60/ 23/ 45.34% XXXXXXXXXXXXXXXXXXXXXXX
 66/ 36/ 51.13% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 72/ 32/ 56.27% XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
 78/ 13/ 58.36% XXXXXXXXXXXXX
 84/ 20/ 61.58% XXXXXXXXXXXXXXXXXXX
 90/ 28/ 66.08% XXXXXXXXXXXXXXXXXXXXXXXXXXXX
 96/ 24/ 69.94% XXXXXXXXXXXXXXXXXXXXXXXX
102/ 28/ 74.44% XXXXXXXXXXXXXXXXXXXXXXXXXXXX
108/ 11/ 76.21% XXXXXXXXXXX
114/ 10/ 77.81% XXXXXXXXXX
120/ 16/ 80.39% XXXXXXXXXXXXXXXX
126/ 13/ 82.48% XXXXXXXXXXXXX
132/ 23/ 86.17% XXXXXXXXXXXXXXXXXXXXXXX
138/ 17/ 88.91% XXXXXXXXXXXXXXXXX
144/ 12/ 90.84% XXXXXXXXXXXX
150/  9/ 92.28% XXXXXXXXX
156/ 13/ 94.37% XXXXXXXXXXXXX
162/  7/ 95.50% XXXXXXX
168/  1/ 95.66% X
174/ 12/ 97.59% XXXXXXXXXXXX
180/  8/ 98.87% XXXXXXXX
186/  4/ 99.52% XXXX
192/  0/ 99.52%
198/  1/ 99.68% X
204/  1/ 99.84% X
210/  1/100.00% X
```

Exhibit 3-4 Summary of Incompatibility Frequencies

AGENCIES:	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12	#13	TOTALS	
SIZE OF SAMPLE:															
LINES:	40936	30545	44285	513533	116397	101449	7749	115202	64166	25109	229566	24381	57074	1370392	
PGMS:	26	55	66	293	155	187	11	62	43	38	77	20	35	1068	
														WGHTD BY PROGRAM	WGHTD BY AGENCY
PROGRAM SIZE (LINES):															
MEAN:	1574.46	555.36	670.98	1752.67	750.95	542.51	704.45	1858.10	1492.23	660.76	2981.38	1219.05	1630.69	1283.14	1261.05
STDDEV:	1877.15	341.28	487.74	2197.62	969.96	583.04	634.77	1820.12	2168.40	445.12	2358.10	733.84	672.02	1710.43	
PROGRAM AGE (YEARS):															
MEAN:	2.38	5.16	2.64	6.64	9.24	4.69	7.16	6.65	3.27	9.04	6.24	10.58	4.57	6.06	6.02
STDDEV:	1.77	2.58	2.02	3.97	4.26	2.82	3.04	4.59	3.06	3.61	3.64	2.60	2.55	4.00	

—— INCOMPATIBILITY COUNTS ——

INCOMPATIBILITY CODES:	(OCCURRENCES/NUMBER OF AFFECTED PROGRAMS PERCENTAGE AFFECTED%)														
2	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
3	0/ 0	2/ 2	24/ 2	0/ 0	22/ 8	1/ 1	0/ 0	20/ 6	0/ 0	0/ 0	203/ 21	4/ 1	0/ 0	276/ 41	
	0.00%	3.64%	3.03%	0.00%	5.16%	0.53%	0.00%	9.68%	0.00%	0.00%	27.27%	5.00%	0.00%	3.84%	4.18%
4	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
5	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	10/ 10	0/ 0	4/ 4	0/ 0	0/ 0	41/ 41	0/ 0	0/ 0	55/ 55	
	0.00%	0.00%	0.00%	0.00%	0.00%	5.35%	0.00%	6.45%	0.00%	0.00%	53.25%	0.00%	0.00%	5.15%	5.00%
6	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	1/ 1	0/ 0	0/ 0	0/ 0	0/ 0	1/ 1	
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	2.33%	0.00%	0.00%	0.00%	0.00%	0.09%	0.18%
7	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
8	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
9	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
10	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	15/ 9	0/ 0	3/ 3	18/ 12	
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	11.69%	0.00%	8.57%	1.12%	1.56%
11	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
12	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
13	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
14	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
15	0/ 0	0/ 0	0/ 0	715/ 15	11/ 1	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	726/ 16	
	0.00%	0.00%	0.00%	5.12%	0.65%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.50%	0.44%
16	8/ 3	4/ 2	2/ 2	132/ 35	13/ 4	1/ 1	0/ 0	12/ 5	3/ 3	0/ 0	170/ 21	22/ 3	47/ 11	414/ 90	
	11.54%	3.64%	3.03%	11.95%	2.58%	0.53%	0.00%	8.06%	6.98%	0.00%	27.27%	15.00%	31.43%	8.43%	9.39%
17	0/ 0	11/ 7	5/ 1	265/ 15	51/ 10	28/ 7	0/ 0	20/ 3	0/ 0	0/ 0	50/ 9	75/ 7	94/ 5	599/ 64	
	0.00%	12.73%	1.52%	5.12%	6.45%	3.74%	0.00%	4.84%	0.00%	0.00%	11.69%	35.00%	14.29%	5.99%	7.34%
18	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
19	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
20	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
21	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
22	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
23	19/ 11	0/ 0	3/ 3	15/ 4	1/ 1	36/ 11	0/ 0	7/ 1	224/ 18	0/ 0	444/ 54	0/ 0	50/ 25	799/128	
	42.31%	0.00%	4.55%	1.37%	0.65%	5.88%	0.00%	1.61%	41.86%	0.00%	70.13%	0.00%	71.43%	11.99%	18.44%
25	0/ 0	0/ 0	0/ 0	36/ 36	1/ 1	8/ 7	0/ 0	0/ 0	5/ 5	0/ 0	0/ 0	0/ 0	0/ 0	50/ 49	
	0.00%	0.00%	0.00%	12.29%	0.65%	3.74%	0.00%	0.00%	11.63%	0.00%	0.00%	0.00%	0.00%	4.59%	2.18%
26	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
27	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	

Exhibit 3-4 (continued)

28	3/ 2	0/ 0	0/ 0	0/ 0	2/ 2	1/ 1	0/ 0	22/ 9	22/ 9	2/ 2	82/ 30	2/ 2	4/ 3	140/ 60	
	7.69%	0.00%	0.00%	0.00%	1.29%	0.53%	0.00%	14.52%	20.93%	5.26%	38.96%	10.00%	8.57%	5.62%	8.29%
29	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
30	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
31	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
33	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
34	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
35	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
37	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
38	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
39	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
40	0/ 0	3/ 2	0/ 0	0/ 0	2/ 1	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	20/ 5	0/ 0	1/ 1	26/ 9	
	0.00%	3.64%	0.00%	0.00%	0.65%	0.00%	0.00%	0.00%	0.00%	0.00%	6.49%	0.00%	2.86%	0.84%	1.05%
42	0/ 0	0/ 0	0/ 0	0/ 0	1/ 1	0/ 0	0/ 0	4/ 1	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	5/ 2	
	0.00%	0.00%	0.00%	0.00%	0.65%	0.00%	0.00%	1.61%	0.00%	0.00%	0.00%	0.00%	0.00%	0.19%	0.17%
43	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	4/ 1	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	4/ 1	
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.61%	0.00%	0.00%	0.00%	0.00%	0.00%	0.09%	0.12%
44	0/ 0	15/ 5	0/ 0	41/ 8	25/ 8	5/ 2	11/ 2	36/ 21	14/ 2	0/ 0	17/ 5	0/ 0	4/ 3	168/ 56	
	0.00%	9.09%	0.00%	2.73%	5.16%	1.07%	18.18%	33.87%	4.65%	0.00%	6.49%	0.00%	8.57%	5.24%	6.91%
45	0/ 0	0/ 0	0/ 0	0/ 0	5/ 5	0/ 0	0/ 0	24/ 14	5/ 2	0/ 0	0/ 0	0/ 0	0/ 0	34/ 21	
	0.00%	0.00%	0.00%	0.00%	3.23%	0.00%	0.00%	22.58%	4.65%	0.00%	0.00%	0.00%	0.00%	1.97%	2.34%
46	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
47	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	1/ 1	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	1/ 1	
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	1.61%	0.00%	0.00%	0.00%	0.00%	0.00%	0.09%	0.12%
48	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
49	1/ 1	0/ 0	4/ 4	12/ 8	2/ 2	23/ 7	1/ 1	2/ 2	0/ 0	1/ 1	68/ 9	0/ 0	12/ 2	126/ 37	
	3.85%	0.00%	6.06%	2.73%	1.29%	3.74%	9.09%	3.23%	0.00%	2.63%	11.69%	0.00%	5.71%	3.46%	3.85%
50	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
51	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
52	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	1/ 1	0/ 0	0/ 0	0/ 0	0/ 0	1/ 1	
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	2.33%	0.00%	0.00%	0.00%	0.00%	0.09%	0.18%
53	235/ 24	16/ 9	1111/ 50	372/183	567/116	1333/176	82/ 8	347/ 42	580/ 42	211/ 30	691/ 71	9/ 5	245/ 34	5799/790	
	92.31%	16.36%	75.76%	62.46%	74.84%	94.12%	72.73%	67.74%	97.67%	78.95%	92.21%	25.00%	97.14%	73.97%	72.87%
54	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	0/ 0	
55	0/ 0	0/ 0	0/ 0	0/ 0	36/ 35	56/ 41	0/ 0	0/ 0	7/ 7	0/ 0	0/ 0	0/ 0	0/ 0	99/ 83	
	0.00%	0.00%	0.00%	0.00%	22.58%	21.93%	0.00%	0.00%	16.28%	0.00%	0.00%	0.00%	0.00%	7.77%	4.68%

TOTAL (all 55 incompatibilities):

266/ 25	51/ 21	1149/ 54	1588/227	739/125	1502/184	94/ 8	503/ 52	862/ 42	214/ 30	1801/ 77	112/ 14	460/ 35	9341/894	
96.15%	38.18%	81.82%	77.47%	80.65%	98.40%	72.73%	83.87%	97.67%	78.95%	100.00%	70.00%	100.00%	83.71%	82.76%

TOTAL (excluding those no longer incompatible):

263/ 25	35/ 14	1120/ 53	1323/222	662/123	1472/184	94/ 8	441/ 52	840/ 42	212/ 30	1446/ 77	31/ 8	361/ 35	8300/873	
96.15%	25.45%	80.30%	75.77%	79.35%	98.40%	72.73%	83.87%	97.67%	78.95%	100.00%	40.00%	100.00%	81.74%	79.13%

TOTAL (also excluding number 53, DISPLAY):

28/ 14	19/ 7	9/ 9	951/ 82	95/ 49	139/ 64	12/ 3	94/ 38	260/ 31	1/ 1	755/ 72	22/ 3	116/ 30	2501/403	
53.85%	12.73%	13.64%	27.99%	31.61%	34.22%	27.27%	61.29%	72.09%	2.63%	93.51%	15.00%	85.71%	37.73%	40.89%

Note: Frequency counts are omitted for those incompatibilities which could not be detected syntactically: 1, 24, 32, 36, 41.

All other incompatibilities are shown. Subsequent action by ANSC/X3J4 has changed the specifications for numbers 3, 4, 17, 28, and 40 to be compatible with COBOL-74.

Exhibit 3-4 (continued)

SUMMARY OF RESERVED WORDS:

RESERVED WORD	OCCURRENCES
ALPHABET	2
CLASS	24
DAY-OF-WEEK	11
END-READ	7
END-RETURN	2
END-WRITE	2
EXTERNAL	22
FALSE	12
INITIALIZE	32
OTHER	34
REFERENCE	2
STANDARD-2	1
TRUE	17

Exhibit 3-5 Proposed COBOL Revisions - Potential Impact Areas

Impact Areas and Elements	Impacts Evaluated: Quantitative Assessment	Impacts Evaluated: Qualitative Assessment	Not Evaluated: Potential Impacts
I. Procurement			
Compiler Costs			X
II. Operations			
Training		X	
Program Development	X		
Program Maintenance	X		
Management of Programming			X
Documentation of Application Software			X
Processing Efficiency		X	
III. Conversion			
Conversion of Application Software	X		
IV. Automated Functional Area Performance			
Functional Enhancement		X	
Requirements Analysis		X	
Program Life Span		X	
V. General			
Programmer Transferability		X	

Another point about interpretion of the statistics is that the detection of incompatibilties was done by a syntactic scan of the source code. Where the incompatibility involves a syntactic change (e.g., the deletion of ENTER), this is a reliable procedure. In those cases where the semantics are being changed or clarified (EXIT PROGRAM closing out PERFORMs, e.g.), however, the best that can be done is to look for source code where such a change might make a difference. This analysis represents, therefore, a worst-case estimate, and is not necessarily realistic. The DISPLAY incompatibility (number 53) is an especially striking example of this. Syntactically, we counted every single occurrence of the DISPLAY verb as an incompatibility, even though the great majority of vendors currently implement this verb as described in the proposed revision. See Appendix D for details concerning each incompatibility.

The age of programs was determined simply by the contents (if any) of the DATE-WRITTEN paragraph. This is, of course, not a foolproof metric. Moreover, it is not as easy as might be supposed to formulate a precise definition of program age (e.g., how much does a program have to be altered before we would say that the old program has not merely been modified, but has been replaced by a new, different program?). Nonetheless, we feel the data is worth presenting, and it does agree with a previous GAO estimate. We were able to find a DATE-WRITTEN entry in 58% of the sample programs.

3.2.3 Application Program Conversion And Maintenance -

In the current setting, the source code for application programs is updated for a variety of reasons:

1. Conversion to a new or modified host system (hardware or software)

2. Accommodating modified functional requirements

3. Correction of errors detected in the code

4. Re-programming to reduce the number of compilers used or to improve processing efficiency

There are no specific data on the updating attributable to each of the above causes; only general statistics on the average life cycle of a program are available.

The interviews with Federal ADP managers revealed that COBOL programs are recompiled at least once a year because of maintenance activities. In some installations, programs are recompiled as many as 6 times annually. A reasonable average is 2-3 times per year.

3.3 Impact Areas

FIPS cost-benefit impact areas are ADP-related activities in which resources and/or outputs are potentially affected by the proposed COBOL revisions. For this analysis, five major areas of ADP activities were examined: Procurement, Operations, Conversion, Automated Function Performance, and General. The 12 individual impact elements within the 5 areas that were analyzed in this study are listed in Exhibit 3-5. The impact elements fall into three categories: quantitative, qualitative, and potential. The type of analysis associated with each of the impact elements is designated by an "X" in the appropriate column of Exhibit 3-5.

Of the 12 impact elements listed in Exhibit 3-5, it was possible to assess three quantitatively: Program Development, Program Maintenance, and Program Conversion. For these impact elements, we were able to construct rough cost-benefit models, and, importantly, adequate data was available.

Of the remaining impact elements, six were assessed qualitatively and three were identified for their potential (but not measured) impacts. Qualitative arguments are provided for: Training, Processing Efficiency, Functional Enhancement, Requirements Analysis, Program Life Span, and Programmer Transferability. For these impact elements, there were strong subjective inferences or plausible arguments that some aspect of an impact can occur. These impact estimates, though imprecise, represent potentially valuable and relevant aspects of the cost-benefit effects of the proposed revisions to Standard COBOL.

The remaining three elements, Compiler Costs, Management, and Documentation, are identified principally to indicate that they were reviewed for potential impacts. For these elements, there seemed to be no basis for estimating any discernible effects of adoption of a new COBOL standard. They are noted in this analysis because they fill out the set of impact elements that should generally be considered in the cost-benefit evaluation of a high level language standard.

3.4 Quantitative Impacts

Given the limitations of the data and process models that are available to describe the application software life cycle, our formulations will attempt to reflect only a rough, first-order measurement of the impacts. Such secondary effects as interest on cost or benefits as distributed over time would be far less in magnitude than the margin of uncertainty about the primary estimates and so would add little to the analysis.

Exhibit 3-6 Impact Parameters and Factors

PARAMETER			VALUE ESTIMATES (1982)			SOURCE/BASIS OF ESTIMATE
SYMBOL	DESCRIPTION		PESSIMISTIC	MOST LIKELY (MODE)	OPTIMISTIC	
AF	Acceptance factor: % of programmers first using COBOL-8x features per year		5%	10%	15%	Best estimate
PSF	Programming savings factor		2%	5%	10%	Best estimate
MSF	Maintenance savings factor		0.5%	1%	3%	Best estimate
NCS	% of existing programs to be converted to COBOL-8x in conjunction with other updating		80%	70%	60%	Best estimate
Ts	% of code susceptible to automatic translation	s=1:	0%	10%	25%	[FCSC-82]
		s=2:	20%	45%	75%	
		s=3:	65%	75%	90%	
		s=4:	80%	90%	100%	
		s=5:	0%	0%	0%	
MCPRs	Average manual conversion rate (lines/day)	s=1:	15.8	17.4	19.4	[FCSC-82]
		s=2:	25.2	28.0	31.5	
		s=3:	78.8	84.0	90.0	
		s=4:	360.0	387.7	420.0	
		s=5:	12600.0	12600.0	12600.0	
ACPR	Average automatic conversion rate (lines/day)		-	630	-	[FCSC-82]

3.4.1 Program Development -

The impact on the program development process is expected to be positive; that is, the new features are expected to save programmer time when coding and interpreting specifications. However, given typical settings in which either slack time or overtime is present, the productivity gains will often be invisible. The productivity gains are an opportunity benefit which may not necessarily be realizable.

The proposed revised Standard COBOL features which have the potential to enhance programmer productivity include the following:

1. Nested programs provide a facility for segmenting large programs into smaller logical units. This should improve both the reliability of these programs, since access to the data in the program can be controlled, and also their maintainability, since such programs are easier to comprehend.

2. Scope delimiters assist in the generation of structured code. While structured programming is possible within COBOL-74, there are a number of anomalous cases where the structuring has to be done using some artificial means, such as PERFORM, because the ability to group statements directly is not present. The delimiters will make the coding smoother and less error-prone.

3. Reference modification allows the programmer to access any part (substring) of a character field without having to re-define the item. This is a very important capability in string manipulation, which can otherwise be clumsy and error-prone. This capability exists in many high level languages (including FORTRAN 77), but not in COBOL-74.

4. EVALUATE statement incorporates a well-known construct from structured programming practices, the multi-way conditional. In COBOL-74, the only reasonable way to handle such logic is with the GO TO...DEPENDING ON statement, which has all the usual problems associated with use of the GO TO.

5. Other constructs which should prove useful in clearing up previously awkward aspects of COBOL are the ability to: PERFORM routines in-line, set up tables with more than three dimensions, accept as well as generate numbers in edited form, and INITIALIZE the values in tables.

Of the above, we were able to search the sample programs for programming practices in which features (3) and (4) could have been used and would have saved time for the programmer. For

feature (3), we searched for data items defined as PIC X (one character only) with an OCCURS clause. For feature (4), we searched for GO TO...DEPENDING ON. In our sample, roughly 22% of the programs could have employed feature (3) and 5% could have used feature (4).

Feature (1) will be especially useful for organizing large programs. Exhibit 3-3 shows that programs of size greater than 1500 lines of source code account for approximately 65% of all the lines of code (even though they constitute only 25% of all programs).

We note that all COBOL programs can make use of feature (2). Moreover, in the interviews conducted with representatives of the various Federal agencies, this enhancement was the one most often cited as potentially improving programming practice.

Thus, we anticipate that the enhancements to COBOL will apply to some degree to virtually all programs in the Federal inventory. For a considerable percentage of the code, the effect will be quite significant.

The basic equation used to estimate the impact of the revised standard on the development of new COBOL programs is given below:

$$\text{Program Development Impact} = \sum_{i=1}^{10} [(NDP)(PAS)(CSP)(PDT)(PPM)] \times [(PSF)(AF)(i)]$$

Where:

i = years; 1,..., 10

NDP = Number of programmer staff-years

PAS = Programmer annual salary

CSP = % of programmers working on COBOL programs

PDT = % of programmers in program development

PPM = % of ADP staff time coding

PSF = Programming savings factor

AF = Acceptance factor of the revised Standard COBOL

AF and PSF have been assigned values of 10% per year and 5%, respectively, and are shown in Exhibit 3-6.

Applying the most likely values to the above parameters from Exhibits 3-1 and 3-6, yields

$$\text{Program Development Impact} = \sum_{i=1}^{10} [(118{,}000)(\$27{,}800)(0.5)(0.4)(0.2)] \times [(0.05)(0.1)(i)]$$

= $36,100,000 (in 1983 constant dollars for ten years and rounded to the nearest $100,000)

3.4.2 Program Maintenance -

Program maintenance concerns those activities involving correcting, perfecting and adapting existing application software, and currently represents 50-70% of the program life cycle costs [BOEH-81], [GAO-81b], [LISW-81].

The principal way in which the proposed changes to standard COBOL would impact on the maintenance function is by increasing the understandability of COBOL programs, and by reducing the error-prone features of COBOL-74. The enhancements to the language cited above under Program Development apply strongly to Program Maintenance also, since they make it easier to read code as well as to write it. Many of the proposed 50 incompatibility changes are intended to eliminate or clarify certain error-prone or ambiguous features of the current COBOL standard.

Theoretically, the proposed changes (enhancements and incompatibilities) can 1) reduce the incidence of programming errors and thus reduce subsequent corrective efforts, 2) enhance the processing performance of a program and reduce subsequent perfective efforts, and 3) enable a program to be more easily adapted to other ADP environments or to accommodate program changes. Mitigating against any near-term, significant impacts, however, are the staff slack and overtime absorbing conditions noted in the productivity discussion, and the fact that the vast majority of maintenance is for a large inventory of existing programs that will likely be converted at a rather slow pace.

To the extent that one can quantify program maintenance impacts attributable to the proposed changes, they are most likely to be found in an opportunity cost argument for freeing up staff resources. A rough approximation to these potential savings is given by the equation below:

$$\text{Program Maintenance Impact} = \sum_{i=1}^{10} [(NDP)(PAS)(CSP)(PMT)] \times [(MSF)(AF)(i)]$$

Where NDP, PAS, CSP, and AF are the same as in the previous equation,

PMT = % of programmers in program maintenance

MSF = Maintenance savings factor

MSF is given a most likely value of 1%. We note that this formulation assumes that maintenance savings will accrue for only those programs that utilize the proposed new features, which is up to 50% in year 5 for the value chosen for AF.

Applying the values for the parameters from Exhibits 3-1 and 3-6, yields:

$$\text{Program Maintenance Impact} = \sum_{i=1}^{10} [(118{,}000)(\$27{,}800)(0.5)(0.6)] \times [(0.01)(0.1)(i)]$$

$$= \$54{,}100{,}00 \text{ (in 1983 constant dollars for ten years and rounded to the nearest \$100,000)}$$

3.4.3 Program Conversion -

Software conversion is the transformation, without functional change, of computer programs and data elements to a new hardware or software processing environment. The greater the degree of incompatibility between the source and target systems and setting, the more difficult the conversion.

Clearly, there will be an extra cost associated with moving programs from a COBOL-74 compiler to a "COBOL-8x" (this is the name sometimes used to refer to the proposed new standard) compiler insofar as there are incompatibilities between the two. This cost is the object of the quantitative analysis. It is also true, however, that in those cases involving the definition by the proposed revision of features which had been ambiguous or implementation-defined, there will be an associated benefit. This is because future conversions within the COBOL-8x standard will not be vulnerable to different implementation of these features.

Exhibit 3-7 Complexity Classes
(Please see Appendix B for a description of the incompatibilities, by number.)

1. Reprogramming: Software is totally dependent on a source system capability lacking in the target environment. Reprogramming involves redeveloping extensive portions of the application logic because the absent capability is fundamental to the initial design.

   Incompatibilities: None.

2. Major program logic modification: Some of the code is sensitive to the source hardware or operating system, and equivalent features are lacking in the target environment. Such logic modification depends on either familiarity with the application or system software, or extensive documentation.

   Incompatibilities: 7, 11, 20, 22, 29, 41, 42, 43

3. Minor program logic modification: Required changes depend on other portions of the code being changed first and a multi-pass automatic translator is not available. While the changes require programming skill, they do not require a knowledge of the functions performed by the software.

   Incompatibilities: 4, 5, 9, 10, 12, 13, 15, 17, 18, 19, 23, 24, 25, 26, 27, 28, 32, 33, 34, 35, 36, 40, 45, 46, 47, 49, 50, 51, 54

4. Simple syntax translation: This class typically involves translation between different dialects of the same high-level language. A technician can prepare and submit translator jobs without knowledge of programming or the program being converted.

   Incompatibilities: 1, 2, 3, 6, 8, 14, 16, 21, 30, 31, 37, 38, 39, 44, 48, 52, 53, 55

5. Software transference: This class comprises software which does not require any translation, but only a simple transfer from source to target environment and possibly some recompiling and relinking. Translation is not necessary because the source and target operating systems, compilers, character sets, and collating sequences are basically identical.

   Incompatibilities: None.

Programs may be brought into conformance with COBOL-8x in the following ways:

1. recoding for the sole purpose of conforming to the new standard

2. recoding in conjunction with a system conversion to a new host system

3. recoding in conjunction with normal software maintenance requiring re-compilation

4. reprogramming to meet new functional requirements of the application

In assessing the impact of the incompatibilities, it is useful to consider the Federal COBOL inventory as a whole, and to ask how many of these programs will eventually be converted to COBOL-8x (as opposed simply to being left as is until no longer needed), and in which of the four ways above this will occur. The list above is ordered from greatest impact per program to least impact. At one extreme, if a program is converted purely for the sake of conformance, then the entire cost of conversion is attributable to the adoption of the new standard. At the other extreme, if a program is completely re-designed anyway, there is no measurable additional cost in seeing that it conforms to the standard. Midway between these cases would be bringing a program into conformance in conjunction with some other form of updating, be that conversion or maintenance. While there is some extra effort involved, much of the conversion overhead (e.g., re-compilation, re-testing) is "free," in that it would be done even if the two versions of the standard were completely compatible. It is worth recalling that programs are recompiled rather frequently (at least once a year) for routine maintenance, and so there is plenty of opportunity for recoding in category 3.

The cost impact is the additional effort expended in each of the above categories. Based on interviews with Federal agencies, and also on a review of the transition process from COBOL-68 to COBOL-74, we conclude that very few, if any, conversions will be done merely for the sake of conformance. This has never been the practice in the past. Moreover, the introduction of such a practice is neither required nor envisaged by existing or planned FIPS for high level programming languages. This policy is stated explicitly in the three most recently issued FIPS for programming languages (FIPS 21-1 COBOL, FIPS 68 Minimal BASIC and FIPS 69 FORTRAN) wherein it is noted that: "It is not intended that existing programs be rewritten solely for the purpose of conforming to the standard."

Also, the previous experience in making the transition from COBOL-68 to COBOL-74 indicates that installations will continue to maintain the compiler for the previous version of the standard for a considerable time after introduction of the new version.

We conclude, then, that the cost of achieving conformance in categories 1 and 4 is negligible, because virtually no conversion will be done in category 1 and there is no impact on conversion in category 4.

Measurable costs, then, are confined to categories 2 and 3, which we will treat together. The key questions are how many conversions will be done this way (as opposed to category 4 or not being done at all), and how much extra effort will be introduced by the incompatibilities.

The first question, the percentage of programs to be converted, may be approached by noting some of the characteristics of the age of programs (see Exhibit 3-3). The statistics on age allow us to formulate only a rough guess as to the pattern of longevity for the current Federal inventory. Note that the statistics are for the age of existing programs. This age distribution would directly reflect longevity only if we assumed that COBOL programs were being created at a constant rate over the last 15 years or so - clearly not the case. Nonetheless, almost any reasonable model one can develop which assumes an average age of six years for Federal COBOL programs will yield a result no greater than 70-75% for the share of programs which will be converted to COBOL-8x over the next ten years.

For example, let us consider a simple model wherein a given agency decides to adopt the COBOL-8x standard in the following way:

1. The conversion will take place over a two-year period, during which both a COBOL-8x compiler and the current compiler are available.

2. Conversion to the new standard will be done in conjunction with normal maintenance.

3. Existing programs will continue to "die" at the normal rate of 15% per year (equivalent to exponential distribution, with average age equal to six years).

4. New programs will be written to conform to the COBOL-8x standard.

Such a model yields a result of 72% of existing code undergoing conversion. Of course this model is highly susceptible to optimization. For instance, the death rate for programs could be accelerated as the normal weighting factors affecting the decision to patch or re-write a given program are tipped in favor of re-writing. In anticipation of the conversion, management guidelines could greatly reduce the incompatibility in new code produced in the year or two preceding the transition period (e.g., no use of ENTER, the new reserved words). Of course this code could not take advantage of the new

features of COBOL-8x, but it would be compatible nonetheless. Or the transition period could be stretched out to three years; this alone would reduce the proportion to 61%.

Next, we must consider the degree of extra effort entailed by the incompatibilities. As part of the effort to develop a model to quantify the conversion cost impacts, we reviewed the conversion models and analyses in [BOEH-81], [DITT-80], [OLIV-79], [FCSC-81] and [FCSC-82]. All these models have similar assumptions and basic structures. They use adaptation or adjustment factors, subjective/experience-based impacts, classes of conversion complexity, cost-per-line of code, and various conversion task work breakdowns. For this analysis, we decided to utilize various parts of the Federal Conversion Software Center model [FCSC-82]. Its formulation is exclusively oriented to and based on Federal ADP systems. Also it reflects [LIV-79] and [FCSC-81], and it provides reasonable definitions of the conversion complexity classes and of average conversion cost-per-line of code by class. Through the use of this model, we can express in a precise way the intuitively natural notion that the costliness of a given incompatibility will depend strongly on how frequently the incompatibility is used (as measured by the sample) and how complex a conversion it entails (as reflected in the assignment of incompatibilities to complexity classes, see below).

The basic equation for the cost of source code conversion in the FCSC model is:

$$SD = \sum_{s=1}^{5} SSDs$$

Where:

s = Complexity class (1 = most complex, 5 = least)

SSDs = Staff-days for required for class s.

SD = Total staff-day cost

The SSD for each s is, in turn, evaluated by:

$$SSDs = \frac{(LOCs)(1-Ts)}{MCPRs} + \frac{(LOCs)(Ts)}{ACPR}$$

Where:

LOCs = Lines of code to be converted in class s

Ts = Percentage (expressed as a fraction) of lines of code capable of being correctly translated automatically for class s.

MCPRs = Average manual conversion productivity rate for class s

ACPR = Average automatic conversion productivity rate

Exhibit 3-7 describes the complexity classes. The assignment of incompatibilities to these classes is based on a technical review of the current and proposed standards, and also on information obtained from the various donor agencies regarding the way in which they use certain features. The only parameter which will be affected by the incompatibilities is LOCs. By increasing the complexity of a conversion, an incompatibility will increase LOCs for more complex classes and decrease it correspondingly for less complex classes. It is convenient to manipulate the above model so as to isolate just this crucial piece of information:

$$\text{Incremental Conversion Impact} = \sum_{s=1}^{5} [CLCs - BLCs] \times \left[\frac{(1-Ts)}{MCPRs} + \frac{Ts}{ACPR} \right] \times (TL)$$

Where:

BLCs = Percentage of lines of code in class s, base case

CLCs = Percentage of lines of code in class s, after impact of incompatibilities

$$= (BLCs)\left(1 - \sum_{i=1}^{s-1} FRCi\right) + (FRCs)\left(\sum_{i=s+1}^{5} BLCi\right)$$

Where FRCx is the percentage of all programs forced up to class x by incompatibilities. Note that the first term reflects the loss of programs from class s to more complex classes, and the second term reflects the gain from less complex classes.

TL = Total lines of code to be converted

We will assign values to MCPRs, APCR, and Ts based on cautious estimates derived from [FCSC-82], see Exhibit 3-6. Further, we shall assume an average salary of $27,800, as before, and 213.2 staff-days per year, again from [FCSC-82]. This allows us to derive a cost per line for each conversion class. Also our model should reflect explicitly the judgment about how many lines of code will be converted. Thus, the final form of our model equation is:

$$\text{Incremental Conversion Impact} = (NCS)(NLP)(NCP) \sum_{s=1}^{5} [CLCs - BLCs]\ (CCLs)$$

Where:

BLCs and CLCs are as before,

NCS = Fraction of COBOL programs which will undergo conversion as in category 2 or 3, described above.

NLP = Average number of lines per COBOL program

NCP = Number of COBOL programs in Federal inventory

CCLs = Conversion cost per line for class s

$$= \left[\frac{(1 - Ts)}{MCPRs} + \frac{Ts}{ACPR} \right] (PAS) / (SDY)$$

(Where SDY = 213.2 staff-days per staff-year, and
PAS = $27,800 programmer annual salary)

Personnel costs are an appropriate basis for classes 1-4 because they are labor-intensive activities. They are a rough but reasonable surrogate for machine time and/or amortization rates for the 5th class, because those costs per line are so small regardless of the cost basis.

The determination of values for BLCs can only be an estimate. Note that as one assigns higher percentages to the more complex classes of the base case, the cost impact decreases. Further, the base case no doubt differs for category 2 (conversion) and category 3 (maintenance). We arrived at base case values (See Exhibit 3-8) based on consultation with the Director of the Federal Conversion Support Center and members of the Federal COBOL Task Group, a group of representatives of Federal agencies which use COBOL extensively. The values used represent base case estimates for conversion (as opposed to maintenance) and so are somewhat pessimistic.

Exhibit 3-8 Cost Analysis of Incompatibilities

Complexity Class (s)	BLCs (Base case distrib.)	FRCs (% progs. forced to class s)	CLCs (resulting distrib.)	Change in distrib.	CCLs Cost per line ($)	Impact ($ millions)
1	0%	0%	0%	0%	-6.77	0.0
2	0%	0.2%	0.2%	+0.2%	-2.65	-1.2
3	20%	26.4%	41.1%	+21.1%	-0.54	-25.3
4	75%	53.8%	57.7%	-17.3%	-0.22	+8.4
5	5%	0%	1.0%	-4.0%	-0.01	+0.1

Net Impact: -17.9 ($ million). Detail figures do not add precisely because of rounding.

Impact based on 70% of 250,000 programs at 1270 lines/program = 222.2 million lines.

Values for FRCs were taken from an analysis of the sample programs. We assigned each of the incompatibilities to one of the five complexity classes and then, by successively masking off the effects of the incompatibilities for each class, (from less to most complex) determined the percentage of programs which would be forced to each of the classes. See Exhibit 3-7 for the assignment of incompatibilities to complexity classes. We then assumed that this "forcing up" was independent of, and therefore uniformly distributed over, the base case. This assumption is expressed in the derivation for CLCs, above. The entire conversion cost analysis is summarized in Exhibit 3-8. The result is a total cost of $17.9 million.

There were a few incompatibilities which could not be detected syntactically. A brief discussion of the more important of these is given to complete the impact analysis for conversion.

1. COBOL-8x specifies the semantics for the case in which there is no next executable statement. Presumably, very few programs rely on the implementor-defined semantics (because it is undefined in COBOL-74) for this situation. Since the semantics simply involves where flow of control will go next, it should be relatively easy to change those programs which are affected. Diagnosing the change in behavior may be a significant problem.

2. COBOL-8x allows reading past end of file. Although most programs use AT END, there should be very few, if any, whose algorithm allows reading past end of file. It is easy enough to install a switch to detect this situation, if need be. The major effect (whether for good or ill, an open question) will be in program development, not conversion.

3. COBOL-8x specifies that STOP RUN closes all files. We assume that few programs rely on the implementor-defined semantics of leaving files unclosed. The only effect of this condition is necessarily outside the program, and it is probable that whatever effect is being achieved can be duplicated with job control statements.

3.5 Qualitative Impacts

Several types of impacts which may result from the implementation of the proposed changes to COBOL-74 could not be quantified in this analysis. Lack of adequate data and the intangible characteristics of the impact are the principal reasons. These qualitative impacts are worth noting because they are legitimate considerations in a pre- or post- implementation assessment of a FIPS.

The qualitative impact discussions are presented in their order of appearance in Exhibit 3-5. Each impact is described in terms of its basic characteristics and whether its effects are expected to be positive or negative.

3.5.1 Training -

In this analysis, training costs are the expenditure of resources to teach a staff the new revised Standard COBOL, over and above the teaching costs expected in the Base Case. For the study horizon this impact will be negative; that is, additional costs above the base case level are expected.

Training expenses typically consist of the cost of 1) instructor time and course preparation, 2) facilities, materials, workshops, and 3) trainee time during the course.

For this formulation, the first two items are judged to be insensitive to the proposed changes, and the third item, trainee time, will be used as the surrogate parameter for the training impact. Further, we make the following assumptions:

1. All programmers (including those with prior experience) reporting to a new job require a training course. The revised Standard and Base Case costs for these programmers are essentially the same.

2. Recurring career enhancement training is essentially the same for the revised Standard and Base Case settings.

The only significant additional cost, therefore, will be when programming staff must be trained "out of sequence," i.e., earlier than in the base case because of a conversion to COBOL-8x. We do not expect this impact to be very great, and almost certainly it will be an order of magnitude lower than the quantitatively evaluated impacts discussed above.

3.5.2 Processing Efficiency -

The premise underlying this impact is that, on the average, in a setting where there are many programmers of varying expertise, a high order language such as COBOL can permit efficient use of the computer system resources. The changes proposed are expected to enhance the processing efficiency of programs in COBOL, and the overall effect for this impact area should be positive.

Another possible effect, which would be negative, is the cost of maintaining an extra compiler while undergoing transition to COBOL-8x. Our study survey showed, however, that

installations typically support at least two COBOL compilers anyway. There will be very little, if any, detectible increase in operating costs traceable to adoption of a new standard. In any event, such a cost would be borne by a compiler for any new standard, even one that was fully upward compatible.

3.5.3 Functional Enhancement -

Using a high order language such as COBOL often leads to improvements in the application functions. Frequently fewer programs are needed, data collection is simplified, and manual operations can be eliminated.

To the extent the proposed changes can improve the accuracy of reports by reducing errors and awkward operations, there can be a contribution to the functions supported by COBOL programs that can use these changes. We expect that these impacts will be positive but indirect and slight.

3.5.4 Requirements Analysis -

The process of specifying functional requirements, and then translating those requirements into program specifications for coding, is affected directly by the choice of programming language. To the extent users can communicate their needs fully and verify the programmed results easily, the requirements analysis process becomes more efficient and effective. All procedural software languages are artificial relative to English. The likelihood of the solution being incorrect, or even of the wrong problem being formulated, increases when it is more difficult for the user and programmer to communicate about the requirements. To the extent that a programmer can express the needs of the user with the help of constructs directly available in COBOL, the communication process between the two is enhanced.

The proposed changes are expected to reinforce this process of better communications about the application requirements analysis by reducing the ambiguous and error-prone features of the existing COBOL standard, as well as by introduction of the new features mentioned earlier. This impact will be positive but indirect and slight.

3.5.5 Program Life Span -

A program with a longer effective life span is preferable to one with a shorter life, under the same operating conditions. The benefits accrue from reduced applications programming, and also, if reusable code modules are constructed, from reduced corrective maintenance costs.

The proposed changes should extend the useful life of COBOL programs in two ways. First, by defining previously vendor-dependent features of the language, programs will become more portable, and hence less likely to be discarded when converting to a new hardware system. Second, there are various new language features which enhance the modularity of the language. Modularized programs are more easily maintained; a radical change in one part does not affect the rest of the code and so there is no need to discard the entire structure and re-write from scratch. This impact will be positive but slight, as COBOL programs are already rather long-lived.

3.5.6 Programmer Transferability -

Since the late 1960's, the pool of personnel interested and trained in COBOL has grown dramatically. A greater supply of personnel already familiar with the language in use reduces training and learning time and makes it easier to recruit staff to use COBOL. Given the turnover ratio in the military and civilian sectors, enhanced programmer transferability is clearly a benefit to Federal agencies that use COBOL.

As mentioned earlier, there are many cases where the proposed revision clarifies or defines language features which were ambiguous or implementor-defined in COBOL-74. Thus, the proposed changes will inprove the ability of programmers to migrate from one implementation to another. Of the qualitative effects of the proposed changes, this impact area is the most significant.

3.6 Sensitivity Analysis

The principal objective of a sensitivity analysis is to assess the degree of variation in the cost/benefit impact estimates generated by changes in the study assumptions, and to provide insight into the validity of the study findings (see Exhibit 3-9 for a summary). Therefore, we will discuss in greater depth those assumptions which are most subject to doubt and which affect the outcome most strongly.

3.6.1 Benefits -

The benefits, as is typically the case for standards, are broad but shallow. Estimating the breadth (i.e., scope) of the benefit is relatively simple: clearly the impact extends throughout the use of COBOL in the Federal government. The difficulty is in arriving at a reasonable estimate for the depth: how much good will the new standard do in an "average" Federal agency? We have tried to be cautious in our estimates of the

programming savings factor (PSF) and maintenance savings factor (MSF). The less precise of these is probably MSF. If we assume that MSF is 2%, instead of 1%, the maintenance benefit increases by $54 million. Such a value is well within reason, but cannot be demonstrated with the available data.

The only point to be made beyond this is that the standard, per se, can offer only a potential for savings in program development and maintenance. The realization of that potential depends on several factors, but especially on good DP management practices.

3.6.2 Cost -

We will now examine those assumptions upon which depends the most likely cost estimate of $17.9 million.

The base case distribution for conversions (BLCs) is not directly supported by any available data (see Exhibit 3-8). If we assume that conformance to COBOL-8x will be accomplished in conjunction with normal software maintenance, rather than conversion, we might plausibly conclude that the base case values are 0%, 0%, 45%, 50%, and 5%, i.e. there will be more class 3 complexity because maintenance may well involve functional changes, whereas conversion, by definition, does not. Under this assumption, the cost estimate becomes $13.2 million.

Clearly, the bulk of the cost stems from those incompatibilities which both occur frequently and force a class 3 modification. There are four of these that deserve some individual comment:

1. Deleting MEMORY SIZE from the standard

2. Deleting ENTER from the standard

3. Defining the effect of EXIT PROGRAM on PERFORMs

4. Defining the order of evaluation of subscripts within PERFORMs

Items 3 and 4 cannot reasonably be changed back to the original specification of COBOL-74. They simply define the semantics of two cases which were not described in COBOL-74.

For item number 1, the effect was completely dependent on the implementation in any event; almost all modern systems accept such information as part of their system control language.

For item number 2, it is technically feasible to keep the specifications of COBOL-74. If this were done the cost estimate would shrink to $11.3 million. There would also be, however, an

adverse effect on the benefit side. ENTER was deleted precisely because it encourages the development of code which is error-prone and difficult to maintain. It would take only a 7% reduction of the benefits to cancel out the $6.6 million cost savings.

It is worth noting that in all four cases above, programs depending on the COBOL-74 specification were not guaranteed to be portable by that specification; all four changes are examples of taking aspects of the COBOL-74 standard which were ill-defined (purposely or not) to begin with, and either deleting the feature outright, or simply defining its effect. In none of these cases is a truly well-defined portable feature being affected.

The final issue is what policy Federal agencies will adopt governing coding practices in the years leading up to the actual transition to a COBOL-8x implementation. We have somewhat pessimistically assumed that, as new code replaces discarded programs, it will have the same degree of incompatibility. If, on the other hand, new code under development were monitored for conformance to COBOL-8x, then the effective percentage of code needing actually to undergo conversion (parameter NCS) would shrink from 70% to 50% within a few years. A figure of 50% implies conversion costs of $12.8 million.

Exhibit 3-9 Sensitivity Analysis

		Assume MSF = 1%	Assume MSF = 2%
most likely assumptions	Benefit:	90.2	144.3
	Cost:	-17.9	-17.9
	Net:	72.3	126.4
base case distribution: 0,0,45,50,5	Benefit:	90.2	144.3
	Cost:	-13.2	-13.2
	Net:	77.0	131.1
assume ENTER unchanged, 10% benefit loss	Benefit:	81.2	129.9
	Cost:	-11.3	-11.3
	Net:	69.9	118.6
NCS = 50%	Benefit:	90.2	144.3
	Cost:	-12.8	-12.8
	Net:	77.4	131.5

(All figures in $ millions)

4.0 FINDINGS

4.1 Interpretation

This study shows that the effect of revising the COBOL standard as proposed should not be dramatic, either for good or ill (see Exhibit 4-1). There is a real opportunity to improve certain features of the language which should not be ignored, but the changes will hardly revolutionize COBOL programming in the Federal sector. At the same time, there will be some problems created by incompatibility, but these are not unusual, either in kind or in degree. Nor should it be surprising that the effect is relatively small; the proposed revision is just that: a revision of an existing standard - and not that markedly different from it.

It is important to put the projected costs and benefits into perspective. An effect of $100 million, spread out over ten years, represents 0.3% of the salaries (unadjusted) of Federal programmers over that same period. Also, the incompatibility problem in particular needs to be addressed. There was a virtual consensus among the ADP personnel we interviewed that modifying source code was among the easier aspects of conversion. They had experienced far more difficulty with conversion of data and of job control code. Some agencies actually had to write their own I/O routines, rather than use those of the new system, because of data incompatibility between the old and new systems. When asked what their biggest problem was, most answered, "the lack of documentation." One interviewee characterized this as the problem of "portability of programs between programmers."

4.2 Recommendations

4.2.1 Incompatibility -

There is no need to improve compatibility between the current and proposed versions of COBOL. While there are theoretical problems, the way in which COBOL is actually used in the Federal government renders them relatively minor. The introduction of any further incompatibilities, however, should be subject to careful evaluation, to ensure that their impact is no more adverse than those considered in this study.

It is important that Federal ADP managers be alerted to the advent of COBOL-8x, and its subsequent impact. In particular, ICST will provide guidance concerning the monitoring of new code under development. Such simple steps as avoiding the use of ENTER will have a large payoff later on when agencies undergo conversion to COBOL-8x.

Also, it is important that automatic conversion be accessible to Federal users. When new compilers start reaching the market, concerned Federal agencies should determine whether

automated translators are available. If not, they should seriously consider co-sponsoring the production of such a software tool.

4.2.2 Education -

ICST will make a vigorous effort to see that the Federal government reaps the full benefit of a new COBOL standard. The new features will help Federal users only if they are known and understood by both programmers and managers. ICST will promote correct use of the new features through a variety of media: guidelines, workshops, and the Federal COBOL Task Group.

4.2.3 Timing -

The benefits of revising the COBOL standard are largely associated with the COBOL programs yet to be written. The costs are associated with those which already exist and depend on features unique to COBOL-74. Therefore, the sooner the standard becomes known and adopted, the better. The problems of incompatibility, real as they are, do not justify delaying the ongoing maintenance and improvement of the COBOL language.

Exhibit 4-1 Summary of Impacts for the Proposed COBOL Standard (Including All Incompatibility Changes Proposed)

IMPACT AREA	QUANTITATIVE (1)	QUALITATIVE
Program Development	$36,100,000	
Program Maintenance	$54,100,000	
Program Conversion	($17,900,000)	
Training	-	Negative, Slight
Processing Efficiency	-	Positive, Slight
Functional Enhancement	-	Positive, Slight
Requirements Analysis	-	Positive, Slight
Program Life Cycle	-	Positive, Slight
Programmer Transferability	-	Positive, Moderate
NET TOTAL	$72,300,000	Positive, Slight

(1) Constant 1983 dollars, summed over 1984-1993; all figures rounded to the nearest $100,000. Figures in parentheses are negative.

REFERENCES

[ANS-74] Technical Committee X3J4, American National Standard programming language COBOL, ANS X3.23-1974, ANSI, May, 1974

[ANS-81] Technical Committee X3J4, Draft Proposed Revised X3.23 American National Standard Programming Language COBOL, ANS, Sep. 1981.

[BOEH-76a] Boehm, B. W. "Software Engineering," IEEE Transactions on Computers, Vol. I, C. 25, No. 12, December, 1976.

[BOEH-76b] Boehm, B. W., Brown, J. R., and Lipow, M. "Quantitiative Evaluation of Software Quality", Proceedings of Second International Conference on Software Engineering, 1976.

[BOEH-81] Boehm, B. W. Software Enginering Economics, Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1981.

[BROO-75] Brooks, F. P. The Mythical Man-Month: Essays on Software Engineering, Addison, Wesley, Reading, Massacusetts, 1975.

[BROO-80] Brooks, R. E. "Studying Programmer Behavior Experimentally: The Problems of Proper Methodology", Communications of the ACM, Vol. 23, No. 4, April, 1980

[CHFS-81] Christensen, K., Fitsos, G. P., Smith, C. P. "A Perspective on Software Science", IBM Systems Journal, Vol. 20, No. 4, 1981.

[CROW-79] Crowley, J. D. "The Application Development Process: What's Wrong with It", JDC Associates, Walnut Creek, California, 1979.

[CURT-80] Curtis, B. "Measurement and Experimentation in Software Engineering", Proceedings of the IEEE, Vol. 68, No. 9, 1980

[DITT-80] Dittman, J. T. Transferability Factor Manual, Veterans Administration, Columbia, MD, March 1981.

[DUBN-81] Dubnow, A. "A Point-by-Point Rebuke of Revision Efforts on COBOL Standards", Data Management, December 1981.

[DEUT-75] Deutsch, D., et al. An Assessment of the NBS Project for Federal Standardization of the COBOL Computer Programming Language, NBS Working Note, June 1975.

[DEUT-76] Deutsch, D. Approval of Federal Government COBOL Standards and Software Management: Survey Results, NBS IR 76-1100, NBS, August 1976.

[FCSC-81] FCSC, Review and Analysis of Conversion Cost-Estimating Techniques, GSA/FCSC-81/001, Federal Conversion Support Center, Falls Church, Virginia, April 1981.

[FCSC-82] FCSC, Federal Conversion Support Center Conversion Cost Model (Version 2), GSA, Office of Software Development, Report No. GSA/FCSC-82/001, Falls Church, Virginia, June 1, 1982.

[FIOR-78] Fiorello, M. and Jaffin, S. Costs and Benefits of Federal Automated Data Processing Standards: Guidelines for Analyses and Preliminary Estimating Techniques, Logistics Management Institute, Wash., DC, 1978.

[GAO-77] Comptroller General, Millions in Savings Possible in Converting Programs from one Computer to Another, GAO, FGMSD-77-34, September 15, 1977

[GAO-80] Comptroller General, Continued Use of Costly, Outmoded Computers in Federal Agencies Can Be Avoided, GAO, AFMD-81-9, December 15, 1980

[GAO-81a] Comptroller General, Government-Wide Guidelines and Management Assistance Center Needed to Improve ADP Systems Development, GAO, AFMD-81-20, February 20, 1981.

[GAO-81b] Comptroller General, Federal Agencies' Maintenance of Computer Programs: Expensive and Undermanaged, GAO, AFMD-81-25, February 1981.

[GAO-82] Comptroller General, Improving COBOL Applications Can Recover Significant Computer Resources, GAO, AFMD-82-4, April 1, 1982.

[GRAY-79] Gray, M. G. Computers in the Federal Government: A Compilation of Statistics - 1978, NBS Special Publication 500-46, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, DC 1979.

[GRAY-81] Gray, M. G. An Assessment and Forecast of ADP in the Federal Government, NBS Special Publication 500-79, Institute for Computer Sciences and Technology, National Bureau of Standards, Washington, DC, 1981.

[JONE-77] Jones, C. Program Quality and Prorammer Productivity, IBM, TR 62.764, San Jose, California, January 1977.

[LISW-81] Lientz, B. and Swanson, E. "Problems in Application Software Maintenance", Communications of the ACM, Vol. 24, No. 11, November 1981.

[MANT-81] Mantei, M. "The Effect of Programming Team Structures on Programming Tasks", Communications of the ACM, Vol. 24, No. 3, March 1981.

[MOHA-81] Mohanty, S. N. "Software Cost Estimation: Present and Future", Software - Practice and Experience, Vol. 11, 1981.

[MUNS-78] Munson, J. B. "Software Maintainability: A Practical Concern for Life-Cycle Costs", Proceedings of COMPSAC, November 1978.

[NBS-72] NBS, Common Business Oriented Language - COBOL, NBS, FIPS Pub 21, March 15, 1972.

[NBS-75] NBS, COBOL, NBS, FIPS PUB 21-1, December 1, 1975.

[NBS-81] The Effects of Future Information Processing Technology on The Federal Government ADP Situation, A. D. Little, Inc., General Systems Group, Inc., Aurora Associates, Inc., NBS GCR 81-342, NBS, September 1981.

[NELS-81a] Nelson, D. "Letter to R. Shoor of Computer World", Control Data Corporation, Sunnyvale, California, February 4, 1981.

[NELS-81b] Nelson, D. "Letter to Mr. R. Widmer, TWA," Control Data Corporation, Sunnyvale, Calif., January 27, 1981.

[OSD-81] Office of Software Development, Software Improvement - A Needed Process in the Federal Government, GSA, OSD-81-102, Falls Church, Virginia, June 3, 1981.

[OTTE-79] Ottenstein, L. M. "Quantitative Estimates of Debugging Requirements", IEEE Transactions on Software Engineering, Vol. SE-5, 1979.

[OLIV-79] Oliver, P., Handbook for Estimating Conversion Costs of Large Business Programs, ADPESO, U.S. Navy, Washington, DC, February 1979.

[SHOE-80] Shoemaker, D. Brief Case Studies on the Implementation of FIPS PUB 21-1 Federal Standard COBOL and Problems Associated with ADP Systems Programs and Data Conversions, Shoemaker ADP/Telecommunications Standards Consultant, Woodbridge, VA, November 1980.

[SHOE-81] Shoemaker, D. and Aurora Assoc., Inc. Product Utilization Survey of the Federal Information Processing Standards Program, Aurora Associates, Inc., Washington, D.C., September 1981.

[STEW-81] Stewart, S. "Memorandum - Comments on Proposed ANSI COBOL," Institute for Computer Sciences and Technology, NBS, December 23, 1981.

[YOUR-79] Yourdon, E. The Second Structured Revolution, Yourdon, Inc., New York, 1978.

APPENDIX A

BASIC COST-BENEFIT ANALYSIS METHODOLOGY

The cost-benefit methodology is based on the 1978 preliminary guidelines for impact assessments of standards [FIOR-78].

1. Specifying the Goals and Objectives for the Standard. Identifies which ICST goal(s) and objectives(s) the prospective standard is expected to contribute to and achieve, respectively.

2. Preparing the Standards Definition Statement. Summarizes all the essential information about the standard necessary to conduct the cost-benefit analysis. At a minimum this includes:

   - How the standard will be developed, used and supported.

   - Essential assumptions and information for the cost and benefit estimates submitted.

   - The historical data trail on the evolution of the standard's design and development, and the corresponding impact and cost estimates from the beginning to the formal implementation of the standard. The trail is provided by the chronological sequence of updated definition statements.

   - A basis for a critical review of the standard's objectives and a discussion on how well the proposed design and development process will satisfy them.

   - Descriptions of the areas of high technological risk and cost-benefit uncertainty.

   The statement can also reference selected information in the backup material that documents the cost and benefit estimates. Each time a cost-benefit analysis is prepared, a Standard Definition Statement will also be prepared. In this way, the statement provides a readily

available audit trail and a basis for critical review of the standard.

3. Defining the Base Case. Defines the status quo for those cost-benefit impact dimensions relevant to the standard. The Base Case can be defined for one or several FIPS that are under consideration. The several FIPS case would occur where the impacts from two or more alternatives tend to overlap, and the individual impacts are not readily distinguishable.

4. Selecting the Cost-Benefit Impact Areas. Specifies cost and benefit impact areas relevant to the standard. A standard will require certain costs for development, implementation, validation and maintenance. In addition, there are costs that can be incurred by the Federal ADP installations that utilize the standard. On the benefit side, a standard will typically tend to have its major economic impacts in one of three areas: procurement, operations, or conversion. These impact areas and elements further delineate how a standard contributes to the ICST goals and objectives concerned with achieving economies in ADP procurement, operations, and conversion. Table A-1 gives the list of impact areas.

5. Constructing the Cost-Benefit Analysis Model. Provides the appropriate model form to quantify the cost and benefit impacts expected of the standard. The model is used to estimate the flows of costs and benefits attributable to the standard over time.

6. Obtaining Data. Collects the data to perform the analysis. In addition we expect to utilize the data from the cost-benefit analysis of selected, individual standards.

7. Estimating and Evaluating the Cost-Benefit. Represents the consolidation of the qualitative and quantitative analyses performed up to this point. The basis of the cost-benefit analysis will be the generation of the Base Case parameters and the computation, within specific confidence intervals, of the effect a new standard scenario will have on cost levels. The techniques used to generate the figures comprising the Base Case address the areas of: (1) level of expected resource consumption, (2) Federal ADP installations' unit cost for each resource, and (3) projected impact(s) of the proposed ADP standard. To aid in this process, cost element equations will be used. To determine the net cost-benefit impact of a FIPS, the costs to the Federal Government (specifically ICST) of developing and implementing the FIPS will be defined and incorporated into the analysis.

8. Presenting and Interpreting the Results. Translates the cost-benefits attributable to the FIPS in the context of the Federal Agencies that will likely be impacted by the standard. This step is crucial as it specifies whether actual budget dollars will be changed (such as in procurement impacts) or whether the impacts are productivity changes (such as when a resource is freed up for other activities).

Table A-1

COST-BENEFIT IMPACT ELEMENTS

100 PROCUREMENT
- 101 Hardware
- 102 Software
- 103 Testing
- 104 Planning and Analysis
- 105 Initial Training
- 106 Documentation
- 107 Installation

200 OPERATIONS
- 201 Operating
- 202 System Management
- 203 Maintenance
  - 203.1 Hardware Maintenance
  - 203.2 Software Maintenance
- 204 Programming
- 205 Ongoing Training
- 206 Facilities
- 207 Security Provisions

300 CONVERSION
- 301 Program Transfer
- 302 Retrofit

400 SYSTEM PERFORMANCE
- 401 Computer System
- 402 Applications Software
- 403 System Users

500 COMPUTER MISUSE
- 501 Errors and Omissions
- 502 Computer-related Fraud and Embezzlement
- 503 Privacy Intrusion
- 504 Alteration of Computer Records
- 505 Theft of Computer Information
- 506 Unauthorized Usage
- 507 Denial of Service
- 508 Equipment Damage

APPENDIX B

INCOMPATIBILITIES BETWEEN COBOL-74 AND COBOL-8X

Description of Incompatibilities, by Number

1. Deletion of rule providing for substitution of double characters for a single character.

2. When figurative constant ALL literal is not associated with another data item, its length is that of the string, rather than one.

3. (Withdrawn) Figurative constant ALL literal cannot be associated with a numeric or numeric edited data item.

4. (Withdrawn) The implicit description of the special register DEBUG-ITEM has been changed.

5. MEMORY SIZE clause deleted.

6. The word ALPHABET is now required before alphabet-name in the SPECIAL-NAMES paragraph.

7. The collating sequence for an indexed file is defined as the native sequence in effect at file creation, not that of the PROGRAM COLLATING SEQUENCE clause (previously undefined).

8. The literal within the CURRENCY SIGN clause may not be a figurative constant.

9. The relative key data item specified in the RELATIVE KEY phrase must not contain the PICTURE symbol "P".

10. Each file in a series of files sharing the same physical reel of tape must be created with a uniform labelling convention. A sort or merge file may not be specified in the MULTIPLE FILE TAPE clause.

11. RERUN clause deleted.

12. Files for which the LINAGE clause has been specified must not be opened in EXTEND mode.

13. When a receiving item is a variable-length data item and contains the object of the DEPENDING ON phrase, the maximum length of the item will be used (as opposed to the actual current length).

14. Digit positions specified by PICTURE "P" will be considered to contain zeros only in numeric operations (as opposed to operations involving non-numeric operands).

15. Evaluation of complex conditions takes place in hierarchical order, left to right, and terminates as soon as the truth value is determined (previously undefined).

16. ALPHABETIC class condition accepts lowercase as well as uppercase letters.

17. (Withdrawn) ALTER statement deleted.

18. CANCEL statement closes all files (previously undefined).

19. CLOSE statement cannot contain both the NO REWIND phrase and the REEL/UNIT phrase.

20. KEY phrase of the DISABLE statement deleted.

21. Subscripts within the REMAINDER clause are evaluated after results stored in GIVING operand (previously undefined).

22. KEY phrase of the ENABLE statement deleted.

23. ENTER statement deleted.

24. When there is no next executable statement in a called program, an implicit EXIT PROGRAM is executed (previously undefined).

25. EXIT PROGRAM closes out all PERFORMs in the called program (previously undefined).

26. Order of evaluation of subscripts within INSPECT statement is defined (previously undefined).

27. No two files in a MERGE statement may be specified in the SAME AREA or SAME SORT-MERGE AREA clause. The only files in a MERGE statement that can be specified in the SAME RECORD AREA clause are those in the GIVING phrase.

28. (Withdrawn) The I-O and EXTEND phrase of OPEN cause non-existing files to be created.

29. REVERSED phrase of OPEN deleted.

30. Order of initialization of multiple VARYING identifiers in PERFORM is specified (previously undefined).

31. Within the VARYING...AFTER phrase of the PERFORM statement, identifier-2 (the higher level control variable) is augmented before identifier-5 (the lower level control variable) is set (as opposed to the reverse order).

32. Execution of READ after the at-end condition occurs is now permitted.

33. The INTO phrase of READ cannot be specified unless: (a) the data item of the INTO phrase and all records associated with the file are group items or alphanumeric, or (b) there is only one record description associated with the file.

34. If a RECEIVE gets a partial message, the next RECEIVE must specify the fully qualified queue structure to get the rest of the message (previously undefined).

35. The INTO phrase of RETURN cannot be specified unless: (a) the data item of the INTO phrase and all records associated with the file are group items or alphanumeric, or (b) there is only one record description associated with the file.

36. STOP RUN closes all files (previously undefined).

37. Order of evaluation of subscripts within STRING statement is defined (previously undefined).

38. In the UNSTRING statement, subscripts of delimiting identifiers are evaluated once, immediately before examination of the sending field for delimiters (as opposed to repeated evaluation).

39. The WRITE statement must not contain both the ADVANCING PAGE and END-OF-PAGE phrases.

40. (Withdrawn) Independent segments deleted.

41. Concept of current record pointer changed to file position indicator.

42. For relative or indexed files, an OPEN I-O, followed by WRITE statements, followed by READ NEXT causes the READ to access the first currently existing record, not whichever record was first at the time of the OPEN.

43. If an alternate key of reference is changed by REWRITE to a value between the current and next value in the file, a subsequent READ NEXT will obtain that same record.

44. The following reserved words have been added:

ALPHABET	ALPHABETIC-LOWER	ALPHABETIC-UPPER
ALPHANUMERIC	ALPHANUMERIC-EDITED	ANY
BINARY	CLASS	COMMON
CONTENT	CONTINUE	CONVERTING
DAY-OF-WEEK	END-ADD	END-CALL
END-COMPUTE	END-DELETE	END-DIVIDE
END-EVALUATE	END-IF	END-MULTIPLY
END-PERFORM	END-READ	END-RECEIVE
END-RETURN	END-REWRITE	END-SEARCH
END-START	END-STRING	END-SUBTRACT
END-UNSTRING	END-WRITE	EVALUATE
EXTERNAL	FALSE	GLOBAL
INITIALIZE	NUMERIC-EDITED	ORDER
OTHER	PACKED-DECIMAL	PADDING
PURGE	REFERENCE	REPLACE
STANDARD-2	TEST	THEN
TRUE		

45. New I-O status values have been added.

46. New communication status key values have been added.

47. Certain error conditions have been defined for exponentiation (previously undefined).

48. When a COPY statement spans several lines, the whole statement is processed, even if some continuation lines are marked as debugging lines.

49. Order of evaluation of subscripts within PERFORM statement is defined (previously undefined).

50. STOP eliminates from the queue any message partially received by the run unit (previously undefined).

51. New communication error key values have been added.

52. COPY in a comment-entry is treated as a comment, not a true COPY statement (previously undefined).

53. After DISPLAY is executed, the output device advances to the beginning of the next line (previously undefined).

54. The description of a data-item referenced in PROCEDURE DIVISION USING ... must not contain a REDEFINES phrase.

55. When the FOOTING phrase is not specified in LINAGE, no footing area exists (previously undefined).

APPENDIX C

SOURCES OF SAMPLE PROGRAMS AND AGENCIES INTERVIEWED

Sample programs were obtained from the following agencies:

1. Department of Agriculture (IBM)
2. Air Force Data Systems and Design Center (Burroughs and Honeywell)
3. Army Materiel Support Systems Agency (IBM)
4. Bureau of the Census (UNIVAC)
5. Computer Systems Command (IBM)
6. Defense Logistics Agency (IBM)
7. Defense Mapping Agency (Burroughs and UNIVAC)
8. Housing and Urban Development (UNIVAC)
9. Bureau of Mines (Burroughs)
10. National Bureau of Standards (UNIVAC)
11. Naval Data Automation Command (IBM)

Interviews were held with the following:

1. George Baird, Director, Federal Software Testing Center and John Caron, Director, Federal Conversion Support Center (both within General Services Administration)

2. Gene Lynd and Kurt Molholm, ADPE Replacement Office, Defense Logistics Agency

3. Steven Merritt, Accounting and Financial Management Division, General Accounting Office

4. James Squier, Office of Information and Management Services, National Oceanic and Atmospheric Administration

5. Kenneth Cammie, Foreign Trade Division, Bureau of the Census

6. Robert Ridgely, Office of Systems Integration, Social Security Administration

7. Edward Crim and Roland Crenwelge, Information Storage and Retrieval Division, Defense Mapping Agency

8. John Roach, Joan Sullivan, and Kurt Kroeger, Management Systems Division, National Bureau of Standards

9. James Flaherty and Lisa Pershing, System Support Division, Internal Revenue Service

Questions for COBOL 8X Impact Study

The objective of this analysis is to determine the costs and benefits to the Federal ADP community if the COBOL-8X prospective FIPS is adopted. There are two fundamental dimensions we need to understand in order to assess the potential impacts:

1. The current base case setting regarding the investment, utilization, and associated costs of COBOL-related activities in the Federal ADP community

2. The specific ways the prospective COBOL-8X FIPS differs from - and would cause changes to - the current COBOL-related activities.

Questions of interest include:

1. How many COBOL programs are there in the Federal inventory? What is the average COBOL program lifetime? What is the annual rate of growth of the COBOL inventory?

2. What is the distribution of COBOL programs by compiler type (e.g., FIPS 21, FIPS 21-1, etc.)? What % of the COBOL programs are in full compliance with COBOL standards? Which standards?

3. What is the average turnover rate (replacement rate) for COBOL compilers? How does it compare with the turnover rate of hardware systems?

4. What is the distribution of the number of COBOL compilers per ADP installation? What is the extra effort involved to maintain an additional compiler?

5. How many of the items, proposed to be changed in COBOL 8X, are used currently? What is the frequency of use per COBOL program? What is the distribution?

6. How will the proposed changes affect current programming and maintenance practices? Are there reasonable implications for productivity, reduction of errors, documentation, processing performance, etc.?

7. How will the COBOL-8X changes affect future conversions? Will they increase the conversion effort? Will the impact be different for different manufacturers' systems?

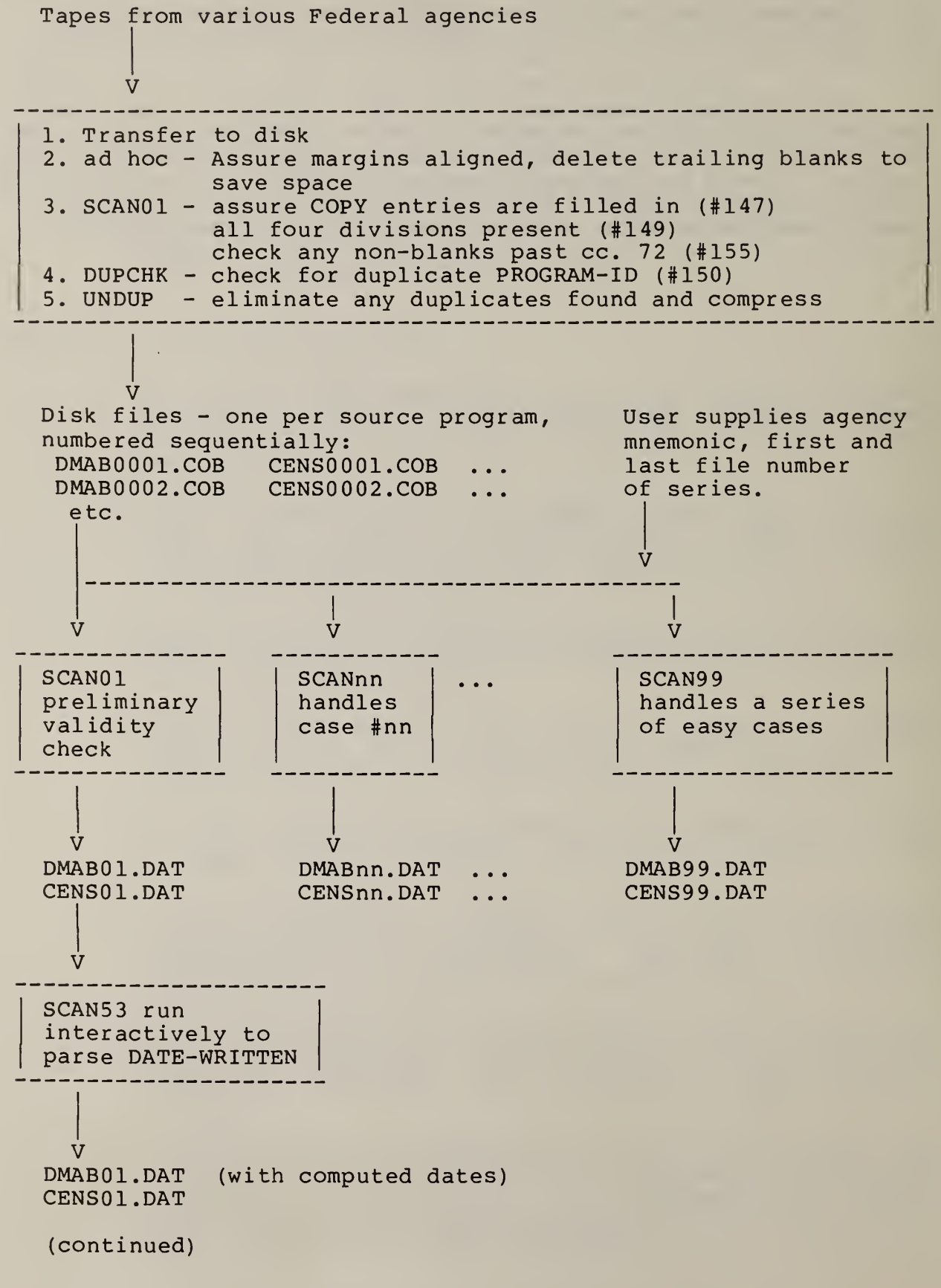
8. What are the major problems with the development and maintenance of applications programs - Programmer turnover, high error rates, poor documentation, etc.? Will the proposed COBOL 8X changes have a positive or negative affect on these problems?

APPENDIX D

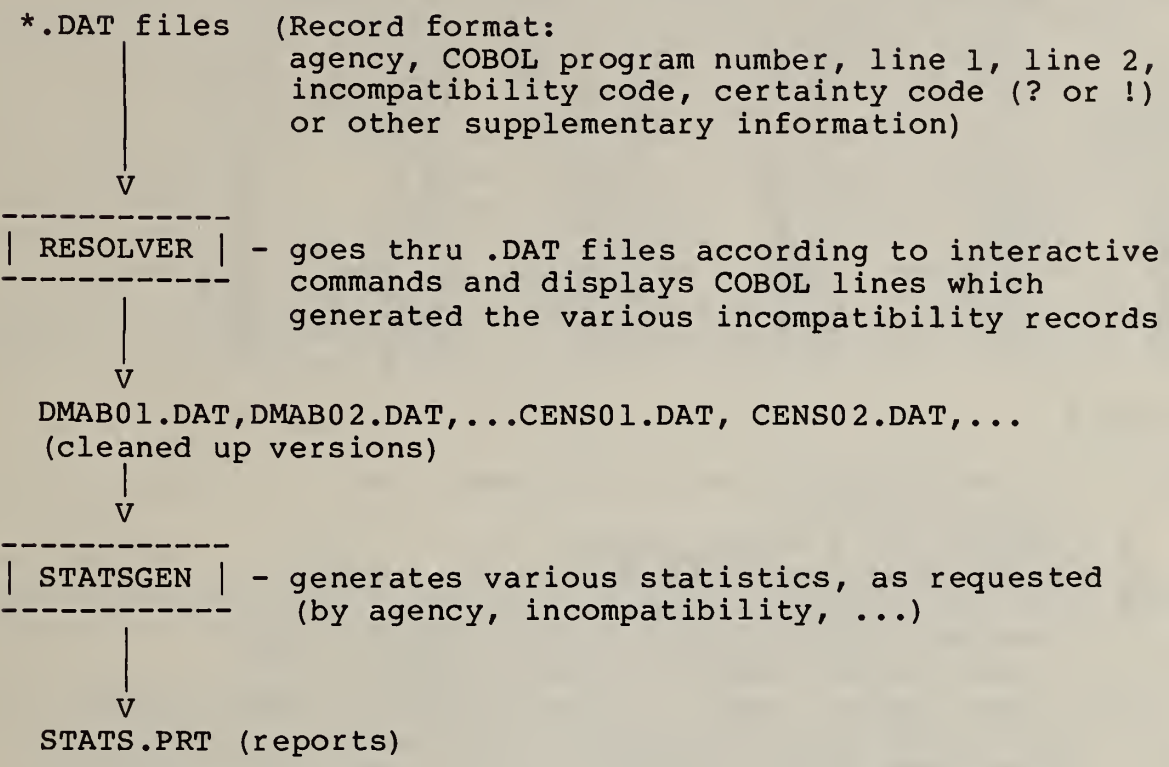
SYSTEM DESIGN FOR THE ANALYSIS OF SAMPLE COBOL PROGRAMS

This appendix summarizes the system which was set up to detect and count occurrences of the various incompatibilities. Although the system is largely automated, human intervention was either allowed or required at certain stages. In particular the RESOLVER program makes it possible to view the actual portions of source code which generated the recording of an incompatibility.

System Design for COBOL Incompatibility Sampling



(continued)



Limitations:

Does not handle/distinguish qualified names - unpredictable results for certain incompatibilities.

Does not analyze CORRESPONDING into elementary operations.

Assumes that a new data-description entry begins on a new line.

Summary of treatment of incompatibilities, along with other features detected for various reasons (see notes).

01 X	11 M	21 M	31 P(30)	41 X	51 M
02 M	12 P(12)	22 M	32 X	42 P(42)	52 M
03cP(03)	13 S	23 M	33 P(33)	43 P(42)	53 M
04cM	14 S	24 X	34 M	44 S	54 S
05 M	15 P(03)	25 M	35 P(33)	45 M	55 P(12)
06 M	16 M	26 S	36 X	46 M	56 U
07 M	17cM	27 S	37 P(37)	47 M	57 N
08 M	18 M	28cM	38 P(37)	48 P(01)	58 N
09 S	19 M	29 M	39 S	49 P(30)	59 U
10 M	20 M	30 P(30)	40cS	50 M	60 U

147-155 E

Key: c - incompatible in Sep. 81 document, but since withdrawn
S - handled by a single program
M - handled by the catchall program, SCAN99
X - not detectible - ignored
P - handled together with another related case (lower number used for program name)
E - these are not actual incompatibilities, but aspects of a program indicating its readiness for processing, plus some global information; all these are handled by SCAN01, which documents their individual meaning
N - these are cases where new features of 8x might profitably be used instead
U - code unused

Program #	handles incompatibilities
1	48,147-155
3	3,15
9	9
12	12,55
13	13
14	14
26	26
27	27
30	30,31,49
33	33,35
37	37,38
39	39
40	40
42	42,43
44	44
54	54
57	57,58
99	2,4-8,10,11,16-23,25,28,29,34,45-47,50-53
not detected:	1,24,32,36,41

Documentation of Scanners for COBOL Impact Study

The following is a short description, for each of the incompatibilities identified in the Draft Proposed Revised X3.23 American National Standard Programming Language COBOL, of the rationale of the approach taken, and the algorithm implemented in the syntactic analysis. It should be noted that the syntactic analysis is not in all cases a foolproof method for finding all incompatibilities. In some cases, incompatibilities cannot be identified at all through syntactic analysis; conversely, in other cases, the analysis can only find an upper bound (worst case) for the number of incompatibilities. Nonetheless, it is plausible to suppose that the analysis yields a roughly reliable measure of the prevalence of features which would be affected by the changes proposed in the draft.

Incompatibility: 1

Rationale: Since the character substitution is implementor-defined, this is a case of implementor-defined syntax (not merely semantics) and thus is undetectible.

Syntactic Analysis: None.

Incompatibility: 2

Rationale: The only place where the 74 standard allows an ALL literal which is not associated with another item is in the SPECIAL-NAMES paragraph.

Syntactic Analysis: There is a search for ALL within SPECIAL-NAMES within ENVIRONMENT DIVISION.

Incompatibility: 3

Rationale: The contexts in which ALL may be associated with a numeric are in a MOVE or a comparison.

Syntactic Analysis: The DATA DIVISION is scanned and a list is constructed of all numeric items, determined as an item containing a PICTURE clause, whose picture-string does not contain A or X. Only elementary names are entered in the list, not the full qualified name. Then, within the PROCEDURE DIVISION, a scan is made for MOVE ALL and for IF, WHEN, or UNTIL (which introduce a condition). For MOVE ALL, all the receiving items are checked against the list, and it's a hit when one is found. For the conditions, they are broken up, if necessary, into the simple conditions connected by AND or OR. Each simple condition is checked for a relation, with all the various

spellings (<, LESS, LESS THAN, etc.) and the comparands are analyzed to see if one is an ALL literal and the other a numeric. Abbreviated conditions are also handled and the "type" of the default comparand (whether ALL or numeric or neither) is saved and applied to later comparands. Thus: 'IF ALL "9" = ALPHA-ITEM OR NUMERIC-ITEM' will be detected as a hit.

Incompatibility: 4

Rationale: The definition of some of the debug items has changed, but not others. Any occurrence of a name whose definition has changed is counted (including group items containing a changed item).

Syntactic Analysis: The PROCEDURE DIVISION is scanned for any occurrence of DEBUG-ITEM, DEBUG-SUB-1, -2, or -3.

Incompatibility: 5

Rationale: Any occurrence is incompatible.

Syntactic Analysis: There is a search for MEMORY within the OBJECT-COMPUTER paragraph within the ENVIRONMENT DIVISION.

Incompatibility: 6

Rationale: This is partially undetectible, since the option 'alphabet-name IS implementor-name' is a hard to isolate. But any of the reserved words unique to the clause are detected.

Syntactic Analysis: There is a search for STANDARD-1, NATIVE, THROUGH, THRU, or ALSO within the SPECIAL-NAMES paragraph within the ENVIRONMENT DIVISION.

Incompatibility: 7

Rationale: Since this is a change in semantics, the best that can be done is to look for cases where it can occur. The only simple way to do this is to detect use of an indexed file, together with the use of the PROGRAM COLLATING SEQUENCE clause. In the absence of this clause, the default is native, and so the new rule that native is used for indexed files would not result in different program behavior. Clearly, this is a worst-case analysis, since very few if any uses of indexed files rely on behavior other than what 8x specifies.

Syntactic Analysis: The OBJECT-COMPUTER paragraph is scanned for

SEQUENCE. If found, a switch is set indicating its presence. Then, there is a search for ORGANIZATION followed by optional IS, followed by INDEXED within the FILE-CONTROL paragraph within the ENVIRONMENT DIVISION. If both INDEXED and SEQUENCE are found, a hit is recorded.

Incompatibility: 8

Rationale: Any occurrence is incompatible.

Syntactic Analysis: Within SPECIAL-NAMES, within ENVIRONMENT DIVISION, there is a search for CURRENCY, then an optional SIGN, then IS, then SPACE, ZERO, QUOTE, HIGH-VALUE, or LOW-VALUE.

Incompatibility: 9

Rationale: Any occurrence is incompatible.

Syntactic Analysis: There is a search for ACCESS, optional MODE, optional IS, any COBOL word, RELATIVE, optional KEY, optional IS, and then the COBOL word which names the data-item, within FILE-CONTROL paragraph. A table of all such data-names is constructed. The DATA DIVISION is then scanned for these items and their picture strings are examined for the occurrence of P. For each one found with PICTURE P, a hit is recorded.

Incompatibility: 10

Rationale: It is impossible to determine the exact conditions called for, so a search is made simply for the occurrence of the MULTIPLE FILE TAPE clause. As is usual when the incompatibility is semantic, this analysis yields a worst-case result, since most such clauses will not actually cause the incompatible behavior.

Syntactic Analysis: There is a search for MULTIPLE FILE within the I-O-CONTROL paragraph within the ENVIRONMENT DIVISION.

Incompatibility: 11

Rationale: Any occurrence is incompatible.

Syntactic Analysis: There is a search for RERUN within the I-O-CONTROL paragraph within the ENVIRONMENT DIVISION.

Incompatibility: 12

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The FILE SECTION of the DATA DIVISION is scanned and each FD is checked for the presence of the LINAGE clause. A table is constructed of all qualifying fd-names. The PROCEDURE DIVISION is then scanned and each OPEN statement is examined to see if any of the LINAGE fd-names are opened in EXTEND mode. The scanner handles the opening of multiple modes and multiple files in one statement. Thus if X is an fd-name with LINAGE, it will be detected in: 'OPEN INPUT A, B EXTEND C, D, X, E, OUTPUT F.'

Incompatibility: 13

Rationale: Although this is a semantic change, it can be isolated syntactically. This is because the difference depends not on the logical flow of the algorithm, but only on the particular structure of the data-item and then its use as a receiving field.

Syntactic Analysis: First the DATA DIVISION is scanned and an array keeps track of the current data structure. The path from root (01 level data-name) to the current data-item is maintained. Whenever a DEPENDING ON clause occurs, all containing items are saved and associated with the DEPENDING ON variable, since their lengths depend on it. Then the program is re-scanned and on this second pass of the DATA DIVISION, once again the current path within the data structure is maintained. A search is made for the names of variables which were the object of the DEPENDING ON clause (i.e., for those items upon which thse length of other data-items depend). When one is found, a check is made for all the group items containing it, to see whether the length of any of them depend on it (as determined by the table built during the first pass). If any such is found, then it is put into a second table, which therefore contains exactly those items which both depend on a given variable for their length and also contain that variable. Finally, the PROCEDURE DIVISION is scanned and the verbs ACCEPT, RECEIVE...INTO, RETURN...INTO, READ...INTO, MOVE, and UNSTRING examined to see whether their receiving items are among those in the second table.

Incompatibility: 14

Rationale: It's not completely clear just which cases constitute an incompatibility. The view taken is that whenever a PICTURE P item is used and associated with a non-numeric item or appears in a context where non-numeric operands are allowed, the behavior may be different and so is incompatible.

Syntactic Analysis: First, the DATA DIVISION is scanned and two lists are constructed: first a list of all PICTURE P data-items (i.e., any item whose picture string contains at least one P) and second a list of all elementary numeric items, except those in the first list. Then, in the PROCEDURE DIVISION, there is a search for situations, either where a PICTURE P item is associated with a non-numeric item, or where it is not associated with any item, but appears in a context where it will be treated as a non-numeric operand. The first case is covered by examining all MOVEs and conditions (introduced by IF, WHEN, or UNTIL) to see if a PICTURE P item is moved to or from, or compared to, a non-numeric item. Multiple receiving fields in the MOVE are handled, as are complex conditions and abbreviated conditions. The second case is covered by looking for a PICTURE P item as an operand of ACCEPT, DISPLAY, or INSPECT.

Incompatibility: 15

Rationale: Again, it is not clear just what constitutes an incompatibility. We distinguished two cases: first any compound condition, and second, the use of a compound condition in a program which bore traces of use of the DEBUGGING module. In the first case, the only possible difference is that while a program running under the '74 rules might abort by evaluating (unnecessarily) a given component of a condition, the same program, under '8X would keep running. The records generated to denote these cases were flagged with a '?'. In the second case, use of DEBUGGING with ALL REFERENCES might work somewhat differently, since the evaluation of a later component of a condition might trigger a debugging report that would not appear if the evaluation were omitted.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for all conditions (introduced by IF, WHEN, or UNTIL), which are examined for AND or OR. The resulting error file is generated with '?' to indicate a questionable incompatibility. A later routine scans for the use of the word DEBUGGING anywhere within the ENVIRONMENT, DATA, or PROCEDURE DIVISION (except in a comment line or nonnumeric literal), or for D in column 7. Either of these is taken as evidence of debugging, and the records for those programs are changed to indicate a possible incompatibility.

Incompatibility: 16

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the presence of ALPHABETIC.

Incompatibility: 17

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the presence of ALTER.

Incompatibility: 18

Rationale: This is a semantic clarification, so all that can be done is a worst-case search for all places where there might be a different effect. It is not obvious whether most compilers already close files upon CANCEL or not.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the presence of CANCEL.

Incompatibility: 19

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for REEL or UNIT, followed by an optional WITH, followed by NO REWIND.

Incompatibility: 20

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for DISABLE, then OUTPUT or INPUT with optional TERMINAL, then a COBOL word, then an optional WITH, and then KEY.

Incompatibility: 21

Rationale: This is a semantic clarification. The scanner looks for any case of a REMAINDER with a subscripted variable.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for REMAINDER followed by a COBOL word, followed by '('.

Incompatibility: 22

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for

ENABLE, then OUTPUT or INPUT with optional TERMINAL, then a COBOL word, then an optional WITH, and then KEY.

Incompatibility: 23

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the presence of ENTER.

Incompatibility: 24

Rationale: This situation cannot be detected syntactically.

Syntactic Analysis: None.

Incompatibility: 25

Rationale: This is a semantic clarification. The scanner finds all instances of EXIT PROGRAM, which represents an upper bound on the true incompatibilities (different behavior).

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the presence of EXIT PROGRAM.

Incompatibility: 26

Rationale: This is a semantic clarification for behavior that occurs when subscripting is used in conjunction with INSPECT. All such cases are detected.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for INSPECT and then the entire statement is scanned for a '('.

Incompatibility: 27

Rationale: Any occurrence is incompatible.

Syntactic Analysis: Within the I-O-CONTROL paragraph, two entities are constructed. The first is an array of lists. Each list in the array contains the fd-names appearing together in a SAME AREA or SAME SORT or SORT-MERGE AREA clause. Thus, there is a list for each such clause. The second is a list of all fd-names appearing in any SAME RECORD clause. Then the PROCEDURE DIVISION is scanned for MERGE. A list of fd-names is taken from

the MERGE, consisting of its primary operand and all file names between USING and GIVING or OUTPUT PROCEDURE. Each file from this list is compared against the SAME RECORD list and any match is counted as an incompatibility. Then the GIVING file, if any, is added to the list of MERGE fd-names. This complete MERGE list is now compared to each of the lists in the array (i.e. each of the AREA lists). If at least two MERGE files are found in any of these AREA lists, it is counted as an incompatibility.

Incompatibility: 28

Rationale: Any use of OPEN I-O or EXTEND is a potential difference in behavior and so all are flagged.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for any occurrence of I-O or EXTEND.

Incompatibility: 29

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the presence of REVERSED.

Incompatibility: 30

Rationale: This change defines the order of at most three move statements: MOVE F1 TO V1, MOVE F2 TO V2, MOVE F3 TO V3, where Fn is the FROM variable and Vn is the VARYING (or AFTER) variable. The only case where this would make a difference is where one of the Fn's depended on one of the Vn's (other than the one it is being moved to). For instance, MOVE A TO B, MOVE C TO A is order-dependent because the sending field of one depends on the receiving field of the other. Likewise for MOVE A (S) TO B, MOVE C TO S. This dependence is sought by the scanner.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for PERFORM statements with the VARYING option. All the control variables (or literals) are captured. We use the convention that Vn is the VARYING (or AFTER) variable, where n=1,2,3 (1 is the "outer" or most inclusive variable, 3 the innermost). Similarly, the FROM operands are denoted by Fn. The primary name of the Vn's (as opposed to subscripts) is isolated as DVn (for instance, if V2 = 'A (2, X)', DV2 = 'A'). Each Fn is then checked to see if any part of it (the primary or subscripts) depends on any of the DVn's, other than the DVn with the same n. If any such dependency is found, it is counted as an incompatibility.

Incompatibility: 31

Rationale: This change affects the order of two pairs of operations: it switches MOVE F3 TO V3 and ADD B2 TO V2 and also MOVE F2 TO V2 and ADD B1 TO V1, where Vn and Fn are as described in number 30, and Bn is the BY variable. The only cases where this makes a difference is, for the first pair, if F3 depends on V2 or V2 or B2 on V3; for the second pair, if F2 depends on V1, or B1 or V1 on V2. These conditions are identified and counted as incompatibilities when encountered.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for PERFORM statements with the VARYING option. All the control variables (or literals) are captured. We use the convention, as above, that Vn is the VARYING (or AFTER) variable, where n=1,2,3 (1 is the "outer" or most inclusive variable, 3 the innermost). Similarly, the FROM operands are denoted by Fn and the BY operands by Bn. The primary name of the Vn's (as opposed to subscripts) is isolated as DVn (for instance, if V2 = 'A (2, X)', DV2 = 'A'). Then, the following checks are done:

Does:	depend on:
V1	DV2
V2	DV3
B1	DV2
B2	DV3
F2	DV1
F3	DV2

If any such dependency is found, it is counted as an incompatibility.

Incompatibility: 32

Rationale: This is a semantic clarification; since any program containing a READ statement with the AT END clause could be affected, it was not deemed worth searching for, since almost all programs would contain such a statement. The only change, however, would be for a program which would have aborted on AT END; such a program would now keep running (somewhat like number 15).

Syntactic Analysis: None.

Incompatibility: 33

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The FILE SECTION is scanned for all FD's and SD's. For each one, all the 01 entries are examined. A count is kept of the number of 01's. Further, if any of them has a

PICTURE clause and the picture string is numeric (i.e., does not contain either A or X), then a switch is set. If, under an FD or SD, there is more than one 01 and at least one has a numeric PICTURE clause, then that fd-name or sd-name is added to a list of such names. The PROCEDURE DIVISION is then scanned for all occurrences of READ with the INTO option, and the fd-name of the READ compared against the list. If it is found, an incompatibility is recorded.

Incompatibility: 34

Rationale: This is a semantic change; it is very difficult to analyze a program syntactically in a detailed way to see where it would make a difference. Therefore, the scanner simply finds instances of the RECEIVE statement.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the presence of RECEIVE.

Incompatibility: 35

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The analysis is the same as for number 33, except that the PROCEDURE DIVISION is scanned for RETURN, rather than READ.

Incompatibility: 36

Rationale: This is a semantic clarification; since any program containing a STOP RUN statement could be affected, it was not deemed worth searching for, since almost all programs would contain such a statement. It is not clear what detectible effect the new specification might have; presumably the state of the unclosed files might be different.

Syntactic Analysis: None.

Incompatibility: 37

Rationale: This is a semantic clarification of the order in which subscripts are evaluated within a STRING statement. It is difficult to analyze the possible different effects beyond simply noting whether subscripts are present.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the STRING statement. The remaining text of the statement is scanned

for the presence of a '('.

Incompatibility: 38

Rationale: This could affect only those UNSTRING statements for which the operand of the DELIMITED BY clause is subscripted. As usual with semantic clarifications, this represents a worst-case analysis of the incompatibility.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the UNSTRING statement. Each the operands of the DELIMITED BY clause (separated by OR) is isolated in turn, and examined for the presence of a '(' (taken to indicate subscripting).

Incompatibility: 39

Rationale: Any occurrence is incompatible.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the WRITE statement. Within the text of the statement, a search is done both for PAGE and for either EOP or END-OF-PAGE. If both are found it is counted as an incompatibility.

Incompatibility: 40

Rationale: The only case in which the deletion of independent segments could possibly affect the logical behavior of a standard '74 program is when an independent segment executes an ALTER. This change is really semantic, because syntactically, segment numbers greater than 49 are still legal; they are simply not treated as independent. As usual for semantic changes, this is a worst-case analysis, since not all such ALTERs will actually cause different behavior.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for sections with segment numbers of 50 or greater. Within such segments, all ALTERs are flagged as incompatibilities.

Incompatibility: 41

Rationale: This incompatibility has no effect on programs beyond that specified in numbers 42 and 43.

Syntactic Analysis: None.

Incompatibility: 42

Rationale: The only case for which this can have an effect is when a program contains an OPEN I-O, a READ NEXT, and a WRITE statement for the same file. Such a combination does not, of course, guarantee different behavior, which depends on the program's logic flow. All the analysis attempts to do, however, is note the presence of these three statements.

Syntactic Analysis: First, the FILE SECTION is scanned and a table is built. Each entry in the table consists of one fd-name and all associated 01-names. Then, within the PROCEDURE DIVISION, there is a search for OPEN statements, READ statements and WRITE statements. There is also a two-dimensional table maintained, for which the first dimension indicates which file is being checked, and the second dimension, which kind of statement (OPEN, READ, or WRITE). When an OPEN is found, all the files (if any) being opened as I-O are noted by an entry being made in the appropriate slot of the table. When a READ is found, with the NEXT option, an entry for that file is made in the table. When a WRITE is found, the record-name is compared against the table built during the scan of the FILE SECTION, to link it back with the fd-name. Then an entry in the two-dimensional table is made for that file. Whenever any entry is made in the table, a check is done to see if the other two entries for that fd-name have been filled. If so, this signifies the presence of all three statements for that fd-name, and an incompatibility is recorded.

Incompatibility: 43

Rationale: The rationale is exactly the same as for number 42, substituting REWRITE for WRITE.

Syntactic Analysis: Same as for number 42, substituting REWRITE for WRITE.

Incompatibility: 44

Rationale: The presence of one of the new reserved words (other than within a comment-line, a comment-entry as in the IDENTIFICATION DIVISION, or a nonnumeric literal) as a user-defined identifier is an incompatibility. If such a word is used as a keyword of an implementor-defined extension, it is not counted, since such use is neither guaranteed nor forbidden by either the '74 or '8x standard.

Syntactic Analysis: The IDENTIFICATION DIVISION is skipped, because reserved words in the comment-entries are insignificant. The other divisions are scanned for any occurrence of any of the new reserved words (except, of course, within comment-lines, or nonnumeric literals). Each occurrence is counted as an

incompatibility. The scanner also records which word was found. Implementor extensions are deleted ad hoc.

Incompatibility: 45

Rationale: Any use of FILE STATUS is potentially incompatible, although the precise analysis depends on the use of old implementor-defined codes and their relation to the new codes.

Syntactic Analysis: The FILE-CONTROL paragraph within the ENVIRONMENT DIVISION is scanned for any occurrence of STATUS.

Incompatibility: 46

Rationale: Any use of the communication status codes is potentially incompatible, although the precise analysis depends on the use of old implementor-defined codes and their relation to the new codes. All CD's have an associated status key.

Syntactic Analysis: The COMMUNICATION SECTION of the DATA DIVISION is scanned for the occurrence of any CD.

Incompatibility: 47

Rationale: Any occurrence is potentially incompatible; however, a real incompatibility exists only if the operands are not checked as being mathematically valid.

Syntactic Analysis: Within the PROCEDURE DIVISION, any occurrence of " ** " is counted as an incompatibility.

Incompatibility: 48

Rationale: Any such COPY statement is potentially incompatible.

Syntactic Analysis: Whenever a COPY statement is found, a switch is set if the end of the statement does not appear by the end of that line. Subsequent lines are examined until either an end of statement (signified by a "." followed by a space) or a debug indicator ("D" in column 7) is found. If the debug indicator is found, an incompatibility is recorded. If an end of statement is found, the switch is re-set to indicate the end of the COPY statement.

Incompatibility: 49

Rationale: Any subscript appearing within a PERFORM...VARYING statement is potentially incompatible, since its setting could occur any time during the processing of the PERFORM.

Syntactic Analysis: Within the text of all PERFORM...VARYING statements, a search is done for any user-defined word (i.e. not a reserved word), followed by a "(", to indicate subscripting. Any such occurrence is recorded as an incompatibility.

Incompatibility: 50

Rationale: Since virtually all programs contain STOP, this applies to any program doing a RECEIVE. Therefore, the scanner merely finds instances of the RECEIVE statement.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the presence of RECEIVE.

Incompatibility: 51

Rationale: This could potentially affect any program using an output CD, since the error key is associated with such entities.

Syntactic Analysis: Within the COMMUNICATION SECTION, the scanner looks for CD (in the A margin), followed by a COBOL word, optionally followed by FOR, followed by OUTPUT.

Incompatibility: 52

Rationale: The only comment-entries are in the IDENTIFICATION DIVISION. Any occurrence of the reserved word COPY therein is a potential incompatibility. It would actually be incompatible only if used as a COPY statement and not a comment.

Syntactic Analysis: The scanner looks for COPY anywhere before the ENVIRONMENT DIVISION.

Incompatibility: 53

Rationale: Any DISPLAY is potentially incompatible. If, however, the vendor performs the required positioning (which most do), then it would not be a real incompatibility.

Syntactic Analysis: The PROCEDURE DIVISION is scanned for the presence of DISPLAY.

Incompatibility: 54

Rationale: Any such occurrence is incompatible.

Syntactic Analysis: Within the LINKAGE SECTION, a search is done for 01, 77 or 1, followed by a COBOL word, followed by REDEFINES. Such COBOL words are saved in a table. Then the PROCEDURE DIVISION header is examined for the USING phrase, followed by any COBOL words. If any of these COBOL words in the USING phrase match any of those in the table, a hit is recorded.

Incompatibility: 55

Rationale: Any such occurrence is potentially incompatible.

Syntactic Analysis: The FILE SECTION is scanned for the presence of LINAGE. When found, the remainder of the clause is searched for the presence of FOOTING. If FOOTING is not found, a hit is recorded.

*U.S. GOVERNMENT PRINTING OFFICE : 1983 0-380-997/857

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. **BIBLIOGRAPHIC DATA SHEET** (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 83-2639	2. Performing Organ. Report No.	3. Publication Date March 1983

4. TITLE AND SUBTITLE

Cost-Benefit Impact Study on the Adoption of the Draft Proposed Revised X3.23 American National Standard Programming Language COBOL

5. AUTHOR(S)

Marco Fiorello & John Cugini

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234
and Aurora Associates, Inc.

7. Contract/Grant No.

8. Type of Report & Period Covered

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

NBS

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

The purpose of the study is to assess the estimated costs and benefits for the Federal Government which would result from adoption of the proposed revision of American National Standard COBOL as a Federal Information Processing Standard (FIPS). Potential benefits of $90.2-million have been identified, stemming primarily from improved productivity in both the development and maintenance of COBOL programs. Estimated costs of $17.9-million have been identified, arising principally from the effort needed to convert old COBOL programs to the new specification, which is incompatible in some respects with the current specification. In support of the study, we conducted interviews with Federal ADP managers and officials, and also analyzed over one thousand Federal COBOL programs for various syntactic characteristics. The study concludes that the potential benefits of a new standard outweigh the estimated costs.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

COBOL; compatibility of programming language standards; conversion costs for COBOL programs; cost-benefit analysis of COBOL standards; Federal use of COBOL; FIPS for COBOL; standardization of COBOL.

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES

15. Price