23506

PROVISIONAL CHECKLIST FOR SOFTWARE CONVERSION PROJECTS

This checklist was prepared during the GAO study of software conversion in the Federal Government. While this checklist is only an unofficial working document, we feel that it may be useful to persons involved in conversion projects.

The study's report to the Congress, Millions in Savings Possible In Converting Programs From One Computer To Another (GAO Report No. FGMSD-77-34), was published on September 15, 1977.

U.S. GENERAL ACCOUNTING OFFICE
ATTN: FGMSD-ADP
441 G. Street, N.W.
Washington, D.C. 20548

923843 / 090831

INTRODUCTION

This checklist was gleaned from many sources. It indicates briefly some answers and actions which might be needed before, during, and after a software conversion project. For each specific conversion, some of the questions on the list may not be relevant. However, some will apply to each case. The questions are shown in the context of assumed situations.

The two basic categories of conversion considered here are (1) conversion to replacement systems and (2) conversion of programs brought in from other installations (foreign programs). Each category is divided into:

--Questions which should be answered early.
--Questions for planning the project.
--Execution of the project.
--Followup.

CONVERSION TO REPLACEMENT SYSTEMS

A. Questions which should be resolved early:
(Situation: The installation is going to get a replacement hardware system or a new operating system for the same hardware.)

1. Schedule: When must the applications be in production on the new system?
2. Labor: Who will do the work?--in-house staff, staff from other agencies, or contractors? Is there enough in-house staff to do the job on time? Will overtime be necessary? If people from another agency are used, is there enough control over them? Are there funds for contracting? (Contracting is attractive when: a) skilled specialists will convert for a fixed price, b) conversion by a contractor will leave in-house staff free for regular tasks. c) experienced conversion specialists with automated aids can complete the conversion more quickly than in-house staff, and d) little need is foreseen to redesign the programs after they are converted.)
3. Examination of software inventory:

--What can be thrown away, i.e., not converted to the new system at all?
--Would some of it be less trouble to write new than to convert?

--What is the expected useful life of the old software? If part of it will not be needed for long after the new system is installed, then emulation or simulation may be a better approach than software conversion, at least for that short-lived part.

4. Examination of data and files inventory:

   --What can be thrown away?
   --What incompatibilities exist between the old and new systems' file-handling capabilities, e.g., different logical organizations supported?
   --Must bridging software be written to automate the conversion of data and files, or is it available off the shelf?
   --Must much data be manually keypunched to implement applications on the new system?

5. Preparation of Detailed Inventory: A detailed inventory of the software files and documentation to be converted should be prepared. Forms for its collection are shown in Appendix A. This inventory is needed regardless of whether the conversion is done in-house or with contractors.
6. Alternatives to conversion: (a) Exploit other software: Do usable programs already exist elsewhere for the intended new system? That is, can new applications software be acquired for the new system instead of converting the old software? (b) Rewrite completely: The old software may be so bad that it should be thrown away and completely new applications written for the new system.

B. Questions for planning the project once the early questions are answered:
(Situation: A software conversion will be done for a significant part of the present inventory of application software.)

1. What about that contractor? If a contractor is to do the work, the one selected must be experienced in conversion. A specialist in conversion is preferable. Several such specialized contractors will now quote a

fixed price on conversion contracts.

The contract for conversion should usually be fixed-price and include lateness penalties and acceptance testing to demonstrate successful conversion. Cost-plus or best-effort contracts are seldom desirable.

2. What is the timing of the effort? Must the conversion work wait for the new system? If some time is available on a system similar to the intended new system, some useful "pre-con ersion" can be done. (The problem with this approach: the other installation will let you have all the third-shift time you want.) Some preparation of the software to be converted is certainly possible on the present (old) system also, e.g., preparation of test data decks and test runs which must be duplicated on the new system. (The problem with this is finding time to do it.)

   Whether preliminary work is deemed feasible or not, the scheduling of resources must be considered, as follows:

3. What resources are available? Will conversion be done on the first shift, or must it be done on the second or third shift? What access to computing resources will the conversion team have? Can interactive terminals, remote job entry terminals, tape drives, disk packs, and other resources be reserved for the exclusive use of the conversion team?

4. What staff Is available (if in-house staff will do it)? Who will be on the conversion team? Considerations include:

   (a) Will team members be working full time on conversion or not? If possible, full time is better.

   (b) Originators of given programs, if still at the installation, can be required to convert their own creations (i.e., live with their own documentation).

   (c) On what shift will the team be doing the conversion work? Are funds for overtime available?

   (d) What training is available for the conversion team?

--General training on the new system.
--Specific training for conversion and/or on use of productivity aids.

5. What conversion aids can be used? Does the installation have, or can it get, automated aids to ease the conversion? Examples include:

   --COBOL-to-COBOL translators.
   --Assembler-to-COBOL translators.
   --Text manipulation processors.

   Several useful aids for COBOL conversion are distributed by the Navy Programming Languages Section. Federal users may get them free by sending in a blank tape.

6. What information is available? Planning should include a determined effort to capture all possible sources of information from the present situation to avoid the work of rediscovery. (See C-1 below.) Hopefully, some of this material will be examined during the planning stage before decisions are made.

C. Execution of the project:

1. Capture of information: All possible information should be collected about the software to be converted. Types of information include:

   --Memories of people that created and used it:

   - Programers.
   - Systems analysts.
   - Operations.
   - User working level.
   - User management.

   --Environmental:

   - Mainframe and peripherals.
   - Operating system, nominal and release number.
   - System software: compiler versions and release numbers and whether or not locally modified.
   - Can the intended new system read tapes from the old one?

--Software to be converted:

- Source code.
- JCL.

--Documentation:

- Source listings.
- Listings of results with test data.
- Test data sets.
- Flow charts.
- Decision tables.
- Run books.
- I/O layouts.
- User requirements statements.
- Other prose:

  Reports.
  Correspondence, if any.

2. Schedule: Establish a schedule with some definable milestones; get it approved, and stick with it if at all possible.
3. Packaging the Conversion Materials: The programs to be converted must be packaged. Each program may be accompanied by any or all of the materials listed above. These materials must be recorded and organized so that conversion steps can be scheduled and monitored and the materials will not be lost. Clerical help is appropriate and very useful for this work--programers hate it anyway.
4. Do not allow the conversion to be complicated with redesign: Remember that the software was carefully evaluated to decide whether to convert it at all. Therefore, it should not need to be improved before it is running on the new system. This means that the conversion leader should keep the users at arms' length until the conversion is complete. However, user requests or ideas for improvement which occur to the conversion programers can be logged for the followup phase.
5. Documentation: The conversion should include documentation as an ongoing task. If at all possible, documentation should be captured on some sort of machine-readable medium (either computer-readable or word-processing equipment) so that later revisions can be done

easily. Of course, comments embedded in the program(s) are themselves machine-readable documentation and valuable documentation if if done properly. And embedded comments, if present, may need revision as part of the conversion of the program.

5. Testing: Conversion must include testing of each program before it is released to production. It is in this situation that preconversion work done on the old system is most beneficial. Also, the test decks and runs that demonstrate conversion should be kept to test later modifications on the new system. The testing should include both "normal" testing with file comparisons to the test results on the old system, and execution monitoring to demonstrate the % of code that is actually executed during the testing. (At least 75% executed should be the target.)

D. Followup--what to do after converted items are in production:

1. Clean up documentation: Documentation of each item of software should be finished immediately after it is ready for production. The documentation kept should includes results of test runs, ideas for future improvements (either efficiency or added function), and narrative of any problems encountered.
2. Evaluate ideas for redesign, augmentation, or efficiency improvements: User requests and programer ideas for improvement, which were logged during conversion, should be evaluated and, if feasible, implemented after the conversion is done.

CONVERSION OF PROGRAMS FROM OTHER INSTALLATIONS

A. Questions which should be resolved early: (Situation: Using a program or set of programs from another installation is being considered as a means of automating a user function.)

1. What is the schedule? When does the user need the function automated? (If the need is soon enough and strong enough, it will force the use of software that exists already

because there will not be time to develop new software.)

2. <u>What are the criteria for selection of suitable foreign software?</u>

--Does candidate software provide the desired end-user functions? (Those functions are the reason for getting it.)
--Is the candidate a complete package with good production-use history and good documentation? If at all possible, the documentation should be in machine-readable form. If it is used at more than one installation already (e.g., has already been converted once), so much the better.
--Is the candidate already in production use on a configuration that is the same as, or very close to, the receiving system? Especially, it should be in production use on a configuration from the same hardware vendor and, hopefully, with the same operating system from that vendor (nominal and release).
--Where was the candidate developed? It may be either a vendor product or a program developed at another agency. If the former, the "conversion effort" is reduced to installation by the vendor, but the evaluation, selection, and benchmarking must be done carefully.

3. <u>What is the data and files situation?</u> Considerations here include:

--Can the data to be processed by the foreign program(s) be prepared for it in an automated manner, e.g., by a "bridging" program which will copy data out of an old (file) format and rewrite it into the format needed by the new program(s)?
--If the data cannot be prepared automatically, is sufficient data entry ("keypunching") labor available, or can it be contracted for? (This aspect of using foreign software is often overlooked or slighted.)
--What file resources will be needed by the new software and the files it is to process?

4. what labor is available? As before (replacement systems, A-5); also, attempt should be made to locate the original author(s) of the program(s) so they can be consulted if problems arise.

B. Questions for planning the project once the early questions are answered:
(Situation: The foreign software has been selected.)

1. What about that contractor? As before; and in this case if a vendor product is chosen, the vendor should perform installation and testing as well as training for recipient staff.
2. What timing is required? Less critical here since the work is not awaiting the arrival of some new hardware resource.
3. What computing resources are available? As before, although probably not nearly as serious a matter in this case because the volume of work is typically smaller.
4. Staff: As before.
5. Conversion aids: As before.

C. Execution of the project:

1. Capture of information: As before, except modified for the case of software from another installation. In the case of a vendor product, the documentation requirements should be included in the contract, and the interviews should include other customers who have used the same product.
2. Schedule: As before.
3. Do not allow the conversion to be complicated with redesign: As before. It is true that even carefully selected foreign software may require modification to suit the new user's needs. However, such modification should wait until the software is converted to the new user's system.
4. Documentation: As before.
5. Testing: As before.

D. Followup:

1. Clean up documentation: As before.
2. Evaluate operation of the software.

Done especially to identify problem areas, and also to identify worthwhile modifications.

3. Interaction with vendors: In the case of vendor software, there should be continuing interaction with vendors for fixing of problems and for incorporation of enhancements. Such interaction should be specified in the contract by which the software is acquired.

APPENDIX A: INVENTORY DATA SHEETS

The following pages show an example of data sheets that would be needed to inventory an installation's software before converting it, as follows:

A-1 System Summary For Conversion.
A-2 Inventory Summary.
A-3 Information By System Required For The Inventory.
A-4 Information By Program (or Process) For The Inventory.

A-1 SYSTEM SUMMARY FOR CONVERSION

Total number of application systems to be converted ______________

Total number of unique master files to be converted ______________

Total number of unique data bases to be converted ______________

Total number of unique applications, (programs/ processes) as follows:

LANGUAGE OR TYPE	NO. OF PROGRAMS	TOTAL SOURCE LINES
COBOL		
Database application 1/		
Database PLI's 2/		
BASIC		
FORTRAN		
Assembler 3/		
PL/I		
High Level Language 4/		
Stand-alone sorts 5/		
Stand-alone utility 6/		
Any others not mentioned in the above list		

NOTES:
(1) Applications include file queries, report generators, sorts, and file updates.
(2) PLI's are Program Language Interfaces in which the process would be described as having been written in a High Level Language such as COBOL.
(3) Assembler applications include those written with macros.
(4) This category is for the sum of applications in all other High Level Languages.
(5) Sorts include vendor supplied or any other packaged sort utility programs where the process itself is a complete job step.
(6) Utilities include those supplied by vendor and any other packaged utility routines.

A-2 INVENTORY SUMMARY

p. 1 of 2

Organization ________________ ________________________

System or Sub-system ________________________ __________

I. Program Specifications: Complete all applicable entries for the four categories of programs, i.e., programs to be (a) translocated 1/ (b) translated, (c) redesigned, and (d) replaced.

	(a)	(b)	(c)	(d)
Source language				
Source system				
Target language				
Target system				
No. application programs (exclusive of categories below)				
No. common subroutines				
No. programs of internal sort calls				
No. modified sort programs				
No. unmodified standalone sorts				
No. standalone utilities				
No. programs by size:				
Under 1000 lines				
1,000-10,000 lines				
More than 10,000 lines				
Total source lines of application programs and sort exits				
Common subroutines				
User coded macros				

(Note: include file and data definitions; include only once any called or "included" code which is not listed as part of subroutines or macros; include code count of multiple versions if all are to be converted.)

1/ Translocated applies to those programs whose functions can be performed by programs which are already running on the target machine.

II. File Specifications: For each record type (fixed or variable length) enter the number of files which are organized as indicated.

	FIXED	VARIABLE
Sequential tape/disk	______	______
Sequential card/print	______	______
Indexed sequential	______	______
Relative	______	______
Other (specify)		
______	______	______
______	______	______

III. Record Specification: For records pertaining to the files of Section II, specify the following:

Multiple record type files	
Number of files	______
Number of unique formats in above count	______
Number of files containing	
Binary data	______
Packed decimal data	______
Floating point data	______
Bit (Logical) data	______
Recording mode (EBCDIC, ASCII, etc.)	______

A-3 INFORMATION BY SYSTEM REQUIRED FOR THE INVENTORY

System identifier ______________________________

Point of contact (name/phone) ______________________________

Date prepared ______________________________

Complete system flow diagram showing the following:

- All interfaces to other systems
- All processes within the system
- All interfaces within the system
- File flow into the system
- Master files which flow within the system
- All output files (products) of the system
- Unique identifiers for all processes within the system
- Any identifiable sub-systems or modules within the system
- All interfaces to data bases which are uniquely identifiable
- All interfaces to libraries, e.g., COBOL COPY libraries
- All interfaces to "special" system software
- Run frequency of the system
- Identify points of operator intervention or special inputs
- Identify processes or points which provide checkpoint/restarts
- Identify points where PROCLIBs or complex JCL procedures are used

Define any problem areas with the current system.
Define how the "new" system should be different.
State any special maintenance requirements and the relative frequency of changes made to the system; i.e., very stable production system with few changes vs. highly changeable system still under development or user needs still being defined.

A-4 INFORMATION BY PROGRAM (OR PROCESS)

p. 1 of 2

System identifier ____________________

Program/process identifier ____________________

Point of contact (name/phone) ____________________

Date prepared ____________________

Language or process type:

Database PLI[1/] ________ (specify prime computer Language) ________

High level language (specify) ____________________

Assembler language (specify) ____________________

Database application ____________________

Stand-alone sort (specify supplier name) ____________________

Stand-alone utility (name) ____________ (vendor/supplier) ________

Any other process not covered above (specify) ____________________

Number of source statements if applicable ____________________

Total number of files INPUT from other processes/systems ________

For each of the files describe the type as follows:

1 - standard secuential tape
2 - standard sequential disk
3 - standard sequential card
4 - standard sequential print
5 - indexed sequential
6 - relative
7 - direct
8 - special file management (specify)

For each of the files describe the length as follows:

1 - fixed length records whether blocked or unblocked
2 - variable length record with fixed length blocks (4 byte length indicator to give the record length, padded blocks)
3 - variable length records with variable length blocks (4 byte record length indicator and a 4 byte block length indicator)
4 - variable length span which has fixed length blocks and the variable length records are spanned across the blocks.

1/ Programming language interface

A-4 INFORMATION BY PROGRAM (CONT'D) p. 2 of 2

INPUT FILE IDENTIFIER	TYPE	LENGTH	AVG. SIZE(BYTES)

Total number of files OUTPUT to other processers or systems ________

OUTPUT FILE IDENTIFIERS	TYPE	LENGTH	AVG. SIZE(BYTES)

Total number of intermediate files used by the system ____________

Patches exist that are NOT reflected in source _____ how many ______

Type of patch (specify) ____________________________

Identify the operational console runbook for this process __________

Checkpoint/restart capabilities are provided in this program _______

Specify when checkpoints are taken ____________________

Complete test data (all inputs) exist for this program/process ______

Size of test data (specify average number of bytes/file ___________

Percentage of logic of the program executed by the test data ________

JCL libraries are used by this program/process (identify) __________

Overlays are used in this program/process________ How many ________