Enterprise Analyzer Reports

Portability Assessment

Level 1 Qualifications Source

1. **Inventory Report** - A list of all source types in the workspace and projects, including counts of objects and source lines
2. **Unresolved Report** - A list of objects referenced within the source code supplied but the repository does not have the source for the object
3. **Unreferred Report** - Sources describing objects that have no reference to them by other objects
4. **Cross Reference Report** - The Cross-reference Report identifies all application references
5. **System Programs** - List of System Programs invoked from the inventory
6. **List of all Objects in the Project** – List of all objects in the project
7. **Unique List of PDS Members Used in Project** – Unique list of PDS members used in project
8. **Unique Missing Objects** – List of unique missing include files and referenced objects

Level 1 Qualifications Data

9. **Unique List of Tables** - Unique list of tables used in the repository
10. **CRUD Report** - View the data operations each program in the project performs and the data objects on which the programs operate
11. **Unique list of VSAM KSDS and ESDS** – Unique list of ORGRANIZATION IS INDEXED data stores
12. **Inbound Interface Files** – Sequential files that are not CREATED by the application
13. **Outbound Interface Files** – Sequential files that are CREATED by the application but not used in the application
14. **Created and Consumed Files** – created and consumed files
15. **Fixed length to be migrated** – Fixed length files that are going to be migrated
16. **Variable Length Files defined in programs** – variable length files that are going to be migrated
17. **Files with multiple data records** – files with multiple data record types

18. **Types of Data COBOL sequential** – ORGANIZATION IS SEQUENTIAL or ORGANIZATION OS RECORD SEQUENTIAL file declarations
19. **Unique list of Data JCL GDG Base** – Unique list of GDGs with (+1)
20. **Instances of Data JCL GDG Base** – List of GDGs with (+1)
21. **Variable length COBOL** - Variable length files require additional effort to convert. COBOL RECORDS [IS] VARYING
22. **Variable length JCL** – Variable length files require additional effort to convert, Includes RECFM=VB or VBA
23. **File Usage COBOL Create** – Files opened for output
24. **File Usage COBOL Read** – Files opened for input
25. **File Usage COBOL Update** – files opened for update
26. **File Usage JCL CREATE**– Files referenced in a JCL DD statement with Disposition "NEW"
27. **File Usage JCL READ** – Files referenced in a JCL DD statement with Disposition "SHR"
28. **File Usage JCL Update** – Files referenced in a JCL DD statement with Disposition "MOD"
29. **DSNs Referring to a Member in a PDS** – Inventory of all JCL Control Cards by listing all PDS libraries with the members that are used in production by the JCL
30. **Seed Files** – Identify Seed Files, files with DISP of OLD or SHR are needed by the JCL and are therefore Seed Files
31. **List of all files in the project** – List of all files in the project
32. **Record Types** – Multiple 01 levels within a single file definition (FD) entry of a COBOL program
33. **Logical files that use records COMP and S9** – Logical files that use in their record structures COMP and S9

Assessment

34. **Embedded Hex General** – Application list where EBCDIC Hex characters are used
35. **BLL Cells used** – Is Base Locator Logic for cell addressing used, and where
36. **Supported System Programs** – System programs supported in the Micro Focus environment
37. **Unsupported System Programs** – system programs not supported in the Micro Focus environment
38. **Unsupported System Programs and without FDS** – Unsupported System programs that do not have FDS

39. **Conversion Requirements** – Source modules requiring conversion to another format, as in called assembler or Easytrieve modules
40. **Modification of pointers by implicit redefinition** - Modification of group item containing a pointer is unsafe, as the memory occupied by the pointer is treated as an alphanumeric data item. In other words, if you move a non-pointer to a group item containing a pointer, pointer problems could exist
41. **Possible Pointer Modification via CALL Statements and Prototypes** - Call prototypes can be used to validate the pointer parameters passed on CALL statements. So reporting problems on parameters that comply with the prototype can be avoided.
42. **Variable Indexing** – Potential pointer problems stemming from modifying pointers by variable indexing
43. **Constant Reference Modification** – Potential pointer problems stemming from modifying pointers by constant reference

Executive Reports

Application Summary

44. **Lines of Code** - Number of lines of code, plus the number of lines of code in included files and any files they include. Comments and blank lines are not counted.
45. **Program Volumes** - V = N * log2(n), where N is Program Length and n is Vocabulary. Minimum number of bits required to code the program.
46. **Maintainability Index** - MI = 171 - 5.2 * ln (PgmVolume) - 0.23 * ExtCycComp - 16.2 * ln (LOC) + 50 * sin (sqrt (2.46 * CommentLines/SourceLines)), where PgmVolume is Program Volume, ExtCycComp is Extended Cyclomatic Complexity, LOC is Lines of Code, CommentLines is Comment Lines, and SourceLines is Source Lines.
47. **Cyclomatic Complexity** - v(G) = e - n + 2, where v(G) is the cyclomatic complexity of the flow graph (G) for the program in question, e is the number of edges in G, and n is the number of nodes. Quantity of decision logic. The number of linearly independent paths (minimum number of paths to be tested). v(G) = DE + 1, where DE is the number of binary decisions made in the program.
48. **Function Points** - Lines of Code divided by K, where K depends on the language: COBOL=77, Natural=52, PL/I=67, PowerBuilder=24. Estimate of the number of end-user business functions implemented by the program.
49. **3-Maintainability Index** - 3-MI = 171 - 5.2 * ln (PgmVolume) - 0.23 * ExtCycComp - 16.2 * ln (LOC), where PgmVolume is Program Volume, ExtCycComp is Extended Cyclomatic Complexity, and LOC is Lines of Code.
50. **Dead lines** - Number of dead lines in programs and used include files. Dead lines are source lines containing Dead Data Elements or Dead Statements. Also, source lines containing dead constructs.
51. **Recursive Number of source lines with COPY** - Number of lines of source, plus the number of lines of source in included files and any files they include. Comments and blank lines are counted.
52. **Recursive number of comments with COPY** - Number of lines of source containing comments, plus the number of lines of source containing comments in included files and any files they include. Inline comments placed on lines with statements are not counted.
53. **Number 0f Code Lines without COPY** - Number of lines of source code. Included files are not counted. Comments and blank lines are not counted.
54. **Commented Code Ratio** – Ratio of commented code lines to total source lines
55. **Inventory project Details** – List of different files types in application area

Repository Statistics

56. **Unverified Objects** - Objects that have not been verified
57. **Verified with errors objects** – Objects that have at least one unrecognizable construct
58. **Lightly verified objects** – Objects that have syntax that can pass light verification test
59. **Missing Objects** – Objects that are referenced but are not part of repository
60. **Unresolved Report** –
61. **Unreferred Report** – Objects that are in repository but nor referenced by any other objects

Performance Optimization (Drill down for ways to improve COBOL)

62. ALTER Statements
63. Arithmetic statements with different byte sizes
64. CALL statements where the program name does not match
65. Complex arithmetic expressions
66. DIVIDE statement with decimal alignment
67. Do not use CORRESPONDING
68. Do not use ON SIZE ERROR
69. Do not use REMAINDER
70. EVALUATE statements where the item is declared in the linkage section
71. EVALUATE statements where the item is a part of a complex expression
72. EVALUATE statements with a larger number of WHEN condition
73. GO TO statements when paragraph is the procedure unit
74. Initialize statements for multiple variables
75. MOVE CORRESPONDING statements
76. MOVE statements where there could be padding
77. MULTIPLY statement with decimal alignment concerns
78. No GOBACK at end of ENTRY block
79. Numeric data items not 1, 2, 4 or 8 bytes
80. OCCURS DEPENDING ON should be avoided
81. PERFOMR compare the counter to a literal value
82. PERFORM statements that do not reference sections
83. PERFORM statements that overlap with others
84. PERFORM THRU statements
85. Procedure falls through to another Procedure
86. Statements that do not use ending scope delimiters
87. STRING statements
88. UNSTRING statements
89. Use of EXIT PROGRAM

Quality Assessment (Drill down for areas that could complicate a re-factor))

90. ALTER statements
91. Avoid REDEFINES of group items
92. Binary Table Search
93. Block size must be zero
94. Choosing efficient computational data items (Declarations)
95. Choosing efficient computational data items (Statements0
96. Comp field declarations must use a sign
97. Comparison of Direct and Relative Indexing
98. Computational statements
99. Computational statements with constants
100. DCLGEN copybooks used for host variables
101. Declaration of numeric items used as subscripts
102. Do not use SECTIONs within the PROCEDURE DIVISION
103. Do not use SPECIAL-NAMES paragraph
104. EVALUATE conditions with decreasing probabilities
105. EVALUATE must have WHEN OTHER clause
106. Exception and Error Handling
107. Explicit Scope Terminators
108. GO TO statements that target non-EXIT paragraphs
109. GOBACK statement must be in first paragraph
110. GOTO must branch to paragraph with an EXIT statement
111. Group items used as transferred data items containing initialized elementary items
112. IF statements not deeper than 3 levels
113. Initialized elementary items used as transferred data items
114. Large table declarations
115. Level 66 declarations not to be used
116. Level 77 declarations not to be used
117. Level numbers that do not align in multiples of 4
118. Look for MOVE statements where the SIGN could be lost
119. Making exponentiation efficient
120. Misalignment in MF COBOL declarations
121. Missing index or subscript checks
122. Must have ENVIRONMENT DIVISION
123. No Switches
124. Non binary OCCURS DEPENDING ON
125. Optimization of constant and variable items

126. Optimization of duplicate items
127. Optimization of variable length data items
128. Paragraphs with at least 4 nested IF statements
129. RECORD SIZE must be zero for fixed length records
130. Relative indexes
131. Relative subscripts
132. Relative subscripts or indexes
133. Running efficiently with CICS
134. SECTION paragraph must have EXIT statement
135. Serial table SEARCH
136. Subscript declarations not binary of 4 or 8 digits
137. Subscripts instead of indexes
138. Test field values using 88 level items
139. Unitialized data items
140. Use figurative constants in the VALUE clause
141. Variables with at least 3 subscripts