Tech Mahindra

LCaaS

(Legacy Code analysis as Service)

User Guide

Contents

1. LCaaS Overview

LCaaS is a Legacy Code Analysis Tool developed by TechM's Enterprise Architecture team which incorporates the industry best practices needed for reverse engineering and aims at reducing the manual efforts & costs drastically. It helps automate the legacy code analysis like COBOL end-to-end through various introspective/intuitive reports.

LCaaS will be extremely valuable for your customers who intend to modernize their legacy applications in COBOL to say Java or .Net or SAP or any technology of choice. Even for those customers who do not intend to modernize, LCaaS will be valuable in providing in-depth documentation of their legacy applications.

2. LCaaS Reports

LCaaS generates lot of intuitive reports and charts, flow diagrams that help us identifying Size, Complexity, I/O intensity, Dependency and Stability of every components as part of legacy modernization and optimization.

These reports will be helpful in doing technical and functional documentation, technical debt analysis and rule extraction that is carried out as part of reverse engineering projects.

Also the data that are collected from these reports would be helpful to come up with modernization estimation and planning and to devise optimal sequence of modernization as part of transformation roadmap creation.

The various reports/Charts that can be generated by LCaaS are as follows:

1. Master Inventory Report
2. Cross reference Report (XREF)
3. CRUD Report
4. Missing Report
5. Orphan/Dead Report
6. Dead para Report
7. Spider Diagram
8. Process Flow Diagram
9. Detailed Flow chart
10. BRE (Business Rule Extraction) Report

2.1. Master Inventory Report

Master Listing of all components identified within the Legacy system and loaded in code repository and also gives LoC, Commented lines, SLoC, Cyclomatic Complexity, dead lines and dead paras/functionalities with in COBOL programs

S.No	Column Name	Description
1	Component_name	Source component name
2	Component_type	Source component type e.g. COBOL,JCL,PROC ,etc…
3	Application_Name	Application name of the component
4	Loc	Lines of code

S.No	Column Name	Description
5	Commented_lines	Number of a commented lines.
6	Blank_lines	Number of blank/empty lines.
7	Sloc	Number of executable source lines of code. (excluding commented lines and blank lines)
8	Cyclomatic_complexity	Cyclomatic complexity is a software metric used to measure the complexity of a program. These metric, measures independent paths through program source code. This metric was developed by Thomas J. McCabe and it is based on a control flow representation of the program
9	No_of_dead_lines	Number of lines of code, which are not reachable/not executed in the control of a program. The source lines of code that are contributing to the dead paragraphs, are accumulated to come up with no of dead lines.
10	Dead_para_count	Number of para's which are not referenced/used in the program in the control flow path
11	Total_para_count	Total number of paragraphs for each Cobol component.
12	Path	Source file location.

2.2. Cross reference Report

One to one relationships between "Calling" and "Called" components that are loaded in repository. Helps in identifying the program level integration touch points

S.No	Column Name	Description
1	component_name	Calling component Name
2	component_type	Calling component Type e.g. COBOL,JCL,PROC ,etc.
3	calling_app_name	Calling application name
4	called_name	Called component
5	called_type	Called component Name
6	called_app_name	Called component Type e.g. COBOL,Copybook,PROC ,etc.
7	dd_name	Called application name
8	access_mode	File access mode (Read/Write) in Jobs
9	step_name	Job (JCL) Step name
10	file_name	Logical file name (DD name)
11	comments	Additional information

2.3. CRUD Report

List of one to one relationships between Program and DB2 tables and its usage (Create, Report, Update, Delete). Helps in identifying the Database level integration touch points

S.No	Column Name	Description
1	Component_name	Source component name

S.No	Column Name	Description
2	Component_type	Source component type e.g. COBOL,JCL,PROC ,etc…
3	Application_Name	Source file application name.
4	Table	Table name that is accessed by program
5	CRUD	Table operation .e.g. Create, Read, Update, Delete.
6	SQL	Embedded SQL Query used in the program to retrieve data from DB, which is used to understand the purpose of the DB operation, columns impacted and the criteria involved

2.4. Missing component report

List of components that are referred by the components loaded in the LCaaS repository, but have no source code in the LCaaS repository.

S.No	Column Name	Description
1	Component_name	Source component name
2	Component_type	Source component type e.g. COBOL,JCL,PROC ,etc…

2.5. Orphan Report

List of components that are not called/referenced by any other components loaded in LCaaS repository.

If a COBOL program is listed as Orphan, then there could be two possibilities.

The program could be a driver program (Online/CICS transaction trigger program) which will be the trigger program. Therefore, it will not be called from any other components. If the driven transaction program is active in system, then it is not definitely an obsolete or dead program. If it is not active or not executed in last 1 year, then that could be potentially dead.

Other orphan COBOL program could be a COBOL program but neither a driver transaction nor referenced by any other program/JCL, then that could be potentially a dead/obsolete program.

S.No	Column Name	Description
1	Component_name	Source component name
2	Component_type	Source component type e.g. COBOL,JCL,PROC ,etc…

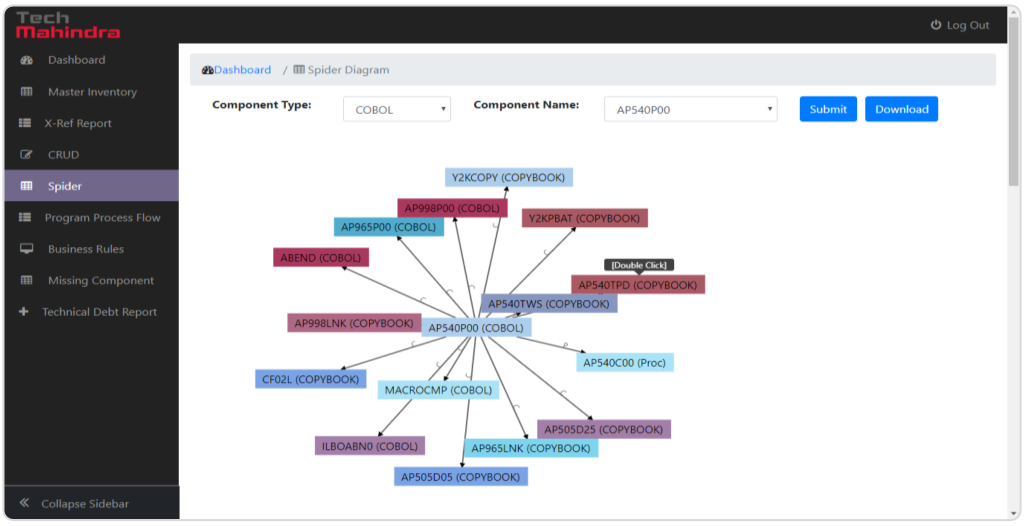
2.6. Dead para report:

List of paragraphs that could be potentially dead, unreachable or unused in nature (when there is a section or paragraph coded in program but respective PERFORM statement is not there in the code, then LCaaS considers it as a dead/unused para).

S.No	Column Name	Description
1	component_name	Source component name
2	component_type	Source component type e.g. COBOL,JCL,PROC ,etc…
3	no_of_dead_lines	Number of lines of code which are not reachable/not executed in the control of a program
4	dead_para_count	Number of para's which are not referenced/used in the program in the control flow path
5	dead_para_list	List of paragraphs that are identified as dead by LCaaS under each Cobol program.

2.7. Spider Flow Chart

Spider chart gives the dependency diagram for a selected component, which gives the Calling and called components relationship in one click and also helps to drill down to any further level from the chart by a **double click** on any specific component of any type in the chart. Thus, this chart helps in carrying out impact analysis of any component and helps to trace both backward (Calling) and forward (Called) impacts.

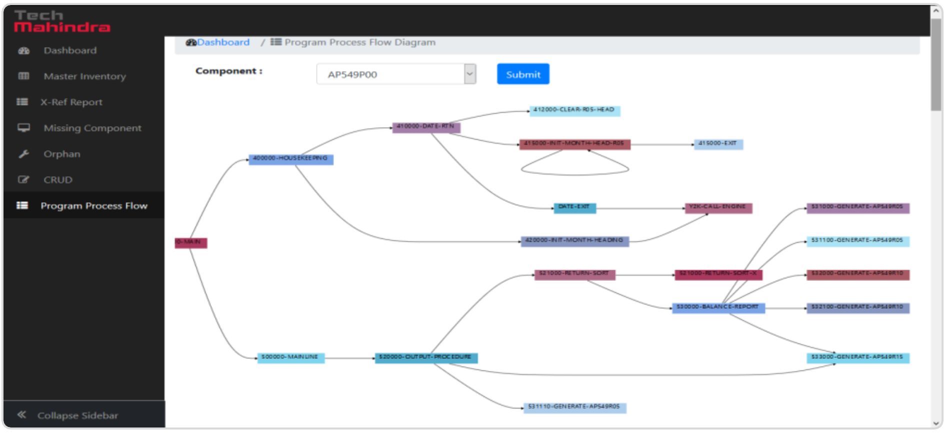


Component Type: COBOL
Component Name: AP540P00

2.8. Program Process Flow Diagram

Program process flow diagram gives the complete control flow at functions level within a COBOL or RPG program.

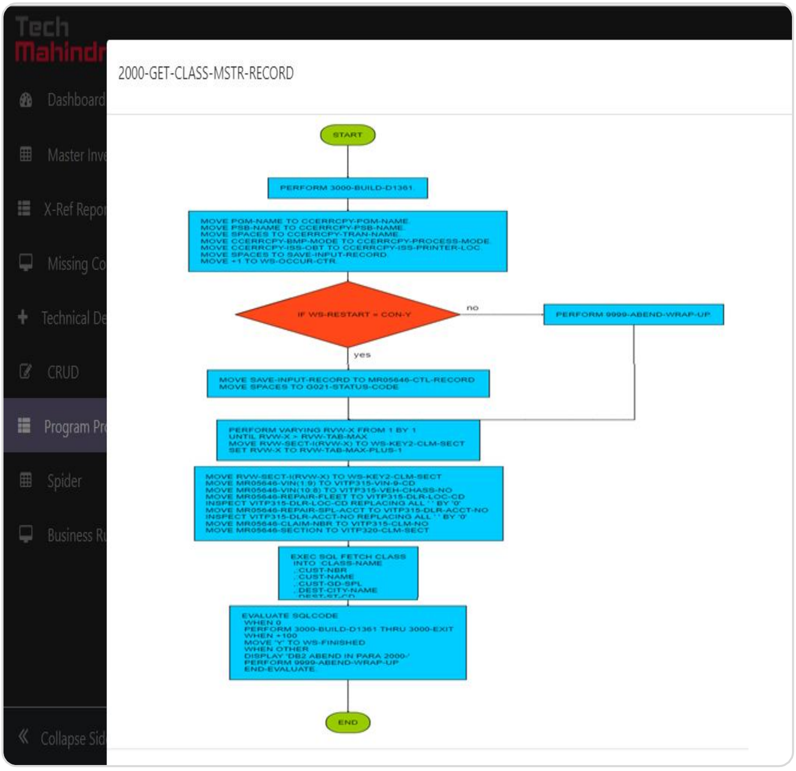
Note: Function refers to Paragraph level in case of COBOL programs and Sub-routine in case of AS400 RPG program



2.9. Program Flow Chart

Program Flow Chart gives a detailed flow diagram of a selected function within Process flow diagram

Note: Function refers to Paragraph level in case of COBOL programs and Sub-routine in case of AS400 RPG program



```
PERFORM 3000-BUILD-D1361
```

```
MOVE PGM-NAME TO CCERRCPY-PGM-NAME
MOVE PSB-NAME TO CCERRCPY-PSB-NAME
MOVE SPACES TO CCERRCPY-TRAN-NAME
MOVE CCERRCPY-BMP-MODE TO CCERRCPY-PROCESS-MODE
MOVE CCERRCPY-ISS-OBT TO CCERRCPY-ISS-PRINTER-LOC
MOVE SPACES TO SAVE-INPUT-RECORD
MOVE +1 TO WS-OCCUR-CTR
```

```
IF WS-RESTART = CON-Y
```

```
PERFORM 9999-ABEND-WRAP-UP
```

```
MOVE SAVE-INPUT-RECORD TO MR05646-CTL-RECORD
MOVE SPACES TO G021-STATUS-CODE
```

```
PERFORM VARYING RVW-X FROM 1 BY 1
UNTIL RVW-X > RVW-TAB-MAX
MOVE RVW-SECT-I(RVW-X) TO WS-KEY2-CLM-SECT
SET RVW-X TO RVW-TAB-MAX-PLUS-1
```

```
MOVE RVW-SECT-I(RVW-X) TO WS-KEY2-CLM-SECT
MOVE MR05646-VIN(1:9) TO VITP315-VIN-9-CD
MOVE MR05646-VIN(10:8) TO VITP315-VEH-CHASS-NO
MOVE MR05646-REPAIR-FLEET TO VITP315-DLR-LOC-CD
INSPECT VITP315-DLR-LOC-CD REPLACING ALL ' ' BY '0'
MOVE MR05646-REPAIR-SPL-ACCT TO VITP315-DLR-ACCT-NO
INSPECT VITP315-DLR-ACCT-NO REPLACING ALL ' ' BY '0'
MOVE MR05646-CLAIM-NBR TO VITP315-CLM-NO
MOVE MR05646-SECTION TO VITP320-CLM-SECT
```

```
EXEC SQL FETCH CLASS
 INTO  CLASS-NAME
  ,:CUST-NBR
  ,:CUST-NAME
  ,:CUST-GD-SPL
  ,:DEST-CITY-NAME
```

```
EVALUATE SQLCODE
 WHEN 0
 PERFORM 3000-BUILD-D1361 THRU 3000-EXIT
 WHEN +100
 MOVE 'Y' TO WS-FINISHED
 WHEN OTHER
 DISPLAY 'DB2 ABEND IN PARA 2000-'
 PERFORM 9999-ABEND-WRAP-UP
 END-EVALUATE
```

2.10. BRE Report

BRE (Business Rules Extraction) report is very crucial report and helps extracting the potential business rules/validations/ technical logics that are coded with in application programs.

The Key features of LCaaS BRE report are as follows:

- Separates the business logic from technical code
- Refactors the code to bring all the logic for a business rule together
- Organizes the business rules into a flow that represents the required sequence of rule execution in the new system.

S.No	Column Name	Description
1	fragment_Id	It organizes the rule into flow that represents the rule sequence to be executed in the target platform.
2	para_name	Name of the para where each fragment id belongs.
3	source_statements	Actual source statement/lines from respective program
4	statement_group	Grouping of each source statements based on the type of operation it performs
5	rule_category	Rule classification of statement grouping.
6	parent_rule_id	Fragment id of the rules that the respective source statement/logic belongs to. Parent rule id the key parameter which helps to bring all the logic for a business rule together