PACBASE TO COBOL, JAVA OR C# MIGRATION

PACBASE is a mainframe based CASE tool which uses a central data-dictionary to hold all definitions. It aimes to by-pass application development using COBOL language.

PACBASE generates COBOL code which is then compiled and deployed. However, the COBOL is not maintainable, and as it stands it does not provide a viable path out of PACBASE.

IBM will drop the support for this product at end of 2015, and is encouraging PACBASE users to move to Rational-Programing-Platform (RPP). RPP can also be used to re-generate the same type of COBOL code, which can then be complied and deployed.

PACBASE TO CLEANED-UP COBOL, CLEANED-UP JAVA OR CLEANED-UP C#

SoftwareMining uses a combination of PACKBASE Segments/Elements, repository and the generated COBOL code as input to our translation tool. The translation tool uses various AI based patterns to cleans-up the code and generate maintainable java or C#. The following is an some of PACBASE specific issues which are taken into consideration to generate maintainabe Java from COBOL code.

- Generated COBOL needs plenty of cleanup: The following areas are some of the main issues contributing to the complexity of generated code:
    - Complicated Flowpath: In many instances, PACBASE COBOL generator goes out of its way to generate complicated code (to avoid re-engineering?). For example an iteration (PERFORM THRU) may be broken down to 3 or 4 different COBOL paragraphs (methods in java), and use GOTO to jump between such paragraphs. Various GOTO removal patterns are used to merge Paragraphs, remove GO TO statements, and cleanup such approaches.
    - Macros: Macro's are used effectively in the same manner as COBOL Copy-Replacing. When used in DATA-DIVISION, different programs may end up with similar structures with different names. SoftwareMining's translation will generate a Data-Access-Object (DAO or java-bean) for each data-structures. However, before generating new DAO, the translator will attempt to identify **and reuse** any previous DAO's with similar structures. We expect between 50% to 70% reduction of in number of structures in the translated application.
    - Obfuscation of names of variable. Whilst the PACBASE data-dictionary contains description of the main structures and variables, the generated Cobol code uses obfuscated names. E.g.

```
MOVE WS-XXX to AXYZ
```

Where AXYZ may be defined in data-dictionary as *Employee Identification Number*. These desciptions are used by SoftwareMining translator to generate a more legible code. For example, the translator can replace references to variable AXYZ with *client.EmployeeIdentificationNumber* (derived from PACBASE SEGMENT information). Furthermore, the translator can work backwards to provide more legible names for variables directly associated with variables with better descritpions! For example, WS-XXX is also regenerated as : ws.EmployeeIdentificationNumber This approach will result in generation of following "maintainable" code:

COBOL:

```
MOVE WS-EMPLOYEE-IDENTIFICATION-NUMBER to EMPLOYEE-IDENTIFICATION-
NUMBER
```

Java:

```
client.setEmployeeIdentificationNumber(ws.getEmployeeIdentificat
ionNumber());
```

C#:

```
client.EmployeeIdentificationNumber =
ws.EmployeeIdentificationNumber;
```

- (For Java / C# migration only) Some PACBASE applications use transaction servers such as CICS or IMS.
  CICS and IMS have their own inherent complexities. For example, in CICS to set Focus on a field, the programmer would have had to assign a value of -1 to a BMS/length attribute: e.g.

  ```
  MOVE -1 to NAMEL
  ```

  SoftwareMining's translation uses a semantic translation of such statements to generate:

  ```
  screen.setFocus("NAME");
  ```

  Such approaches significantly improve legibility and maintainabilitu of the target Java or C# applications.

FINALLY

The above is a very brief summary of the experiences gained. For further information please contact www.softwaremining.com or watch COBOL to Java/C# Project Planning and Management - 10 min Guide with Embedded SQL on Youtube.