Leveraging COBOL 162 (Interworks 2001)

Title: Leveraging legacy applications to create graphical COBOL that may include the Web
Presenter: Charles F. Townsend
Presentation: 162

LegacyJ Corporation
5035 Almaden Expressway
San Jose, CA 95119
Telephone: 408-979-8090 ext. 11
Fax: 408-979-8099
Email: ctowns@legacyj.com

Abstract: *The COBOL programming language has been a very important in expressing business logic. As new technologies are introduced to our business environment it remains important that ties between legacy business applications and new technologies are maintained. This presentation will describe some of the issues to be considered, a few case studies, and summarize some of the items to be considered when bringing together converging technologies.*

Agenda

COBOL FOR LINUX (INTERWORKS 2001) 1

INTRODUCTION 4
BACKGROUND 5
ENVIRONMENT 7
LINUX 9
COBOL CHECKLIST 9
GRAPHICAL USER INTERFACES 11
PORTABLE APPLICATIONS 16
OBJECT ORIENTED 18
ENTERPRISE JAVA BEANS 19

Introduction

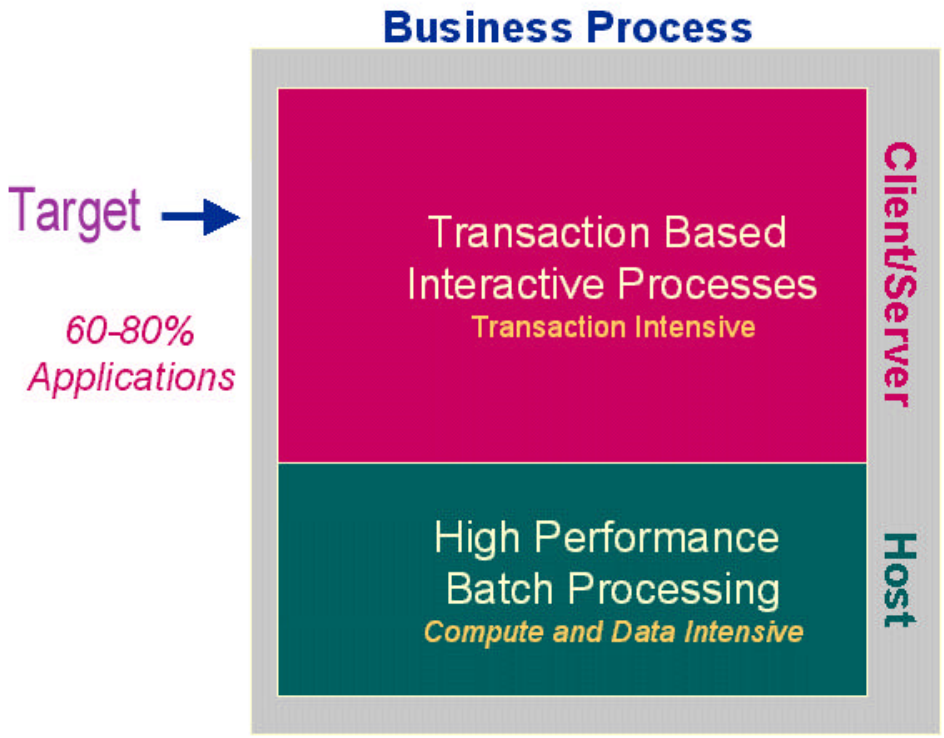
The current business requirement is to build portable software applications that fully exploit legacy business processes and enable future application growth. The source for these business applications may be expressed in many COBOL compiler dialects rooted in disperse and dissimilar operating environments. The challenge is to bring together these disparate COBOL programs and combine them with Java to build enterprise applications.

Background

COBOL holds a unique position as a programming language; ANSI standards and government requirements dictated that a COBOL standard compliant COBOL compiler was available on each computer purchased by the United States Defense Department. Business followed the government lead and found the COBOL syntax ideal for expressing business logic. Consequently, approximately 80% of the world's business applications run on COBOL, and its total worldwide investment exceeds $5 trillion. In short, COBOL serves as the foundation of businesses.

The new dynamic in the market in the past few years is the Internet. Exploiting the Internet is now a major focus of almost every hardware and software vendor such as HP, IBM and Sun Microsystems. Businesses from traditional "brick and mortar" to purely "electronic businesses" have or are adding Internet components to their software enterprise. Federal, state and local government agencies are looking for ways to bring their legacy applications forward and to offer new Internet ready services.

One aspect of this new phenomenon is that, business-to-business (b2b) e-commerce transactions are expected to reach $1.3 trillion by 2003[1]. The challenge is how to move legacy systems forward and not isolate these business applications from the e-commerce components delivered through Java technologies.

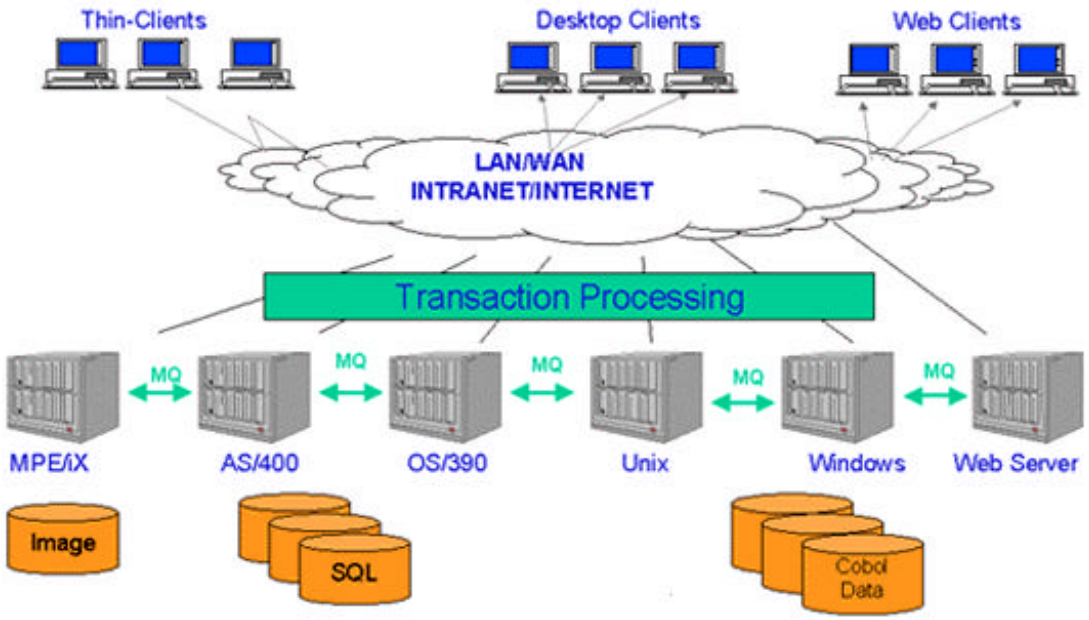


[1] Forrester Research

Environment

It never gets any easier. The complexity of our business environment may never again be comprised of a homogeneous single tier business environment. Components continue to be added to the business environment along with challenges of differing architectures, portable (handheld devices), self-defining data (XML), and differing “net” architectures. The words “cheaper”, “faster” and “stable” are ever present, and Linux is very much part of our business processing paradigm.

The Challenge is to control costs and to Internet enable the business enterprise. Much can be accomplished with the right investment, enough time and resources. Unfortunately, the challenge faced by IT today is to accomplish in days or months that which used to take years. The phenomenal growth of the Internet and e-commerce dictates that business aggressive move into this arena in order to capitalize on its potential. The temptation is to create an Internet solution in isolation and then synchronize with existing legacy application systems. The synchronization task can be complex and costly and without the right tools the challenge might take years and the opportunity is missed.



Linux

The problem with taking business applications written in COBOL to Linux is that there are few options and the options don't really have the same capabilities as are on other operating platforms. Graphical capabilities may be non-existent and features that are common place with other platforms don't seem to exist on Linux.

COBOL checklist

Item	Description
Broad COBOL syntax	Standard COBOL with vendor extensions
COBOL Debugging	Graphical and Line mode debugging
Graphical User Interface	Graphical Screen Section, Graphics in COBOL, Java Interfaces
National Language	DBCS, NLS and numeric internationalization
Object Oriented	OO Directives
Inter language CALL	C, Java
TCP/IP	Networking and NFS capability
SQL Support	SQL Database (Any supported)
Web enabled	HTML interface, Applet, Servlet

Many COBOL vendors don't understand Linux. They don't understand that ease of migration is critical; the compiler and runtime should both understand the legacy environment, but should also understand the new target environment. If it was ????? before it should be once it is on Linux as well, and if I should decide to move up to HP-UX it should work there without any changes to the code.

COBOL vendors sometimes have funny (not so humorous) commercial terms. They want to charge a runtime fee and charge more when I move to larger and larger platforms. The new processing paradigm makes for interesting commercial terms such as charging for usage (clients) connected to the application. Try counting Internet clients. Linux is different and users don't like historic server runtime pricing strategies.

Graphical User Interfaces

Platform independent Graphical COBOL Applications can be as easy as adding graphical elements to a COBOL Screen Section. (Handouts of these samples will be available during the presentation). The following example will generate a graphical COBOL application.

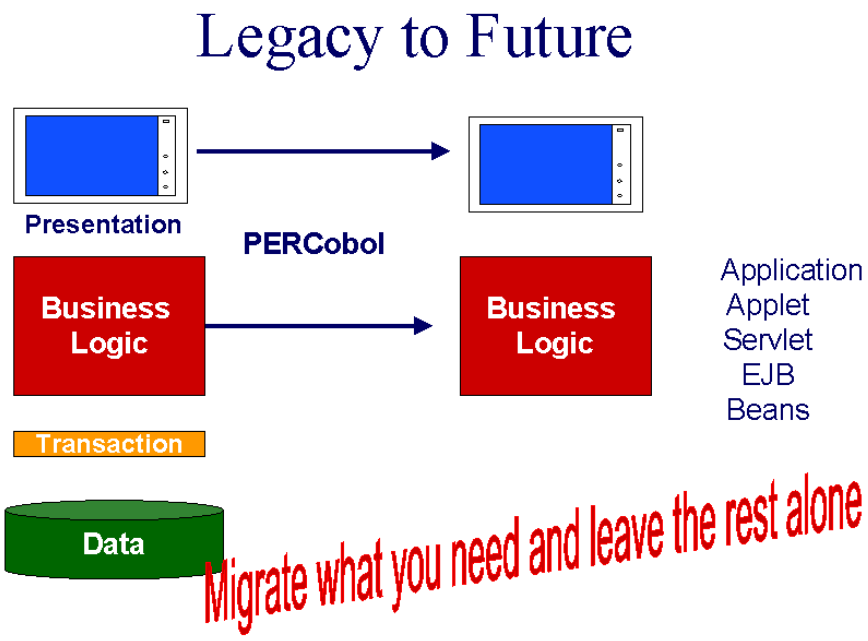
```
 identification division.
 program-id.  sample.

 data division.
 working-storage section.

* these copyfiles are internal, used to define constants
* for graphics support; there is no actual file by these
* names.

 copy "percobol.def".
 copy "percobolgui.def".

* the combo box is automatically sorted, though this may be disabled

* 78 level is for constants; this is common among PC/Unix
* Cobols, though it's supported everywhere in PERCobol.

 78 combo-size value 20.
 78 number-of-combo-choices value 9.

* this is a normal comment
  01  combo-box-choices. | this is an inline comment

      03  pic x(combo-size) value "George W. Bush".
      03  pic x(combo-size) value "Bill Clinton".
      03  pic x(combo-size) value "George Bush".
      03  pic x(combo-size) value "Ronald Reagan".
      03  pic x(combo-size) value "Jimmy Carter".
      03  pic x(combo-size) value "Gerald Ford".
      03  pic x(combo-size) value "Richard Nixon".
      03  pic x(combo-size) value "Lyndon Johnson".
      03  pic x(combo-size) value "John F. Kennedy".
  01  combo-choice
     redefines combo-box-choices
     occurs number-of-combo-choices times
     pic x(combo-size).

  77  frame-text                pic x(200) value
     "PERCobol allows graphics and text in the screen section.
-    "Combo-boxes, radio-buttons, fonts, labels, and entry-fields
-     " are shown.".

  77  check-box-data             pic 9 value zero.
  77  radio-button-data          pic 9 value zero.
  77  entry-data-1               pic x(10).
  77  entry-table occurs 20 times  pic x(70).
  77  combo-data                 pic x(20).
```

```
* the handle type wraps a graphical reference; the handle
* may be typed, as in handle of font below, so the type
* may be used.  the large-font and small-font are built-in
* font references.

 77 large-font handle of font large-font.
 77 small-font handle of font small-font.

* this is a shortcut for definition of crt status, so it
* may be defined inline rather than in the special-names
* section.

 77  key-status is special-names crt status pic 9(4).
       88  exit-button-pushed  value 13. | ascii for return
 screen section.

 01  screen-1.

*    the items in a graphical screen section may have non-integer
*    positions, column and line.

*    the name of the variable (optional, FILLER otherwise)
*    is followed by the control-type (if graphical), such
*    as label or entry-field.  then the property [[=] value]
*    clauses follow.

    03  label "PERCobol Graphical Sample",
        line 1.5, column 21, size 25,
        font large-font, center.
    03  frame, rimmed, font small-font
        line 4, column 4, size 32, lines 9.
     03  label, title frame-text, font small-font,
          line 5, column 5, size 30, lines 7.

    03  label "&Entry Field", line 14, column 5.

    03  entry-field, using entry-data-1
        column + 2, 3-d.

    03  label "&Multi-line Entry Field",
        line + 3, column 5, cline + 2.

    03  entry-field, using multiple entry-table
        line + 1.5, cline + 1, column 8, size 50, lines 5,
        max-lines = 20, vscroll-bar, 3-d,
        no-autosel, use-return.

    03  check-box "&Check-box",
        using check-box-data, line 5, column 38.

    03  frame, lowered,
        line + 1.5, column 37,
        lines 3, csize 28, size 26.

*    radio-button-data is assigned the group-value of the
*    selected radio-button.
```

```
    03  radio-button, "&Left"
        using radio-button-data,
        line + 1, column 38, group-value = 1.

    03  radio-button, "&Right"
        using radio-button-data,
        column + 3, group-value = 2.
    03  label "&Combo-Box" line + 2.5, column 38.

    03  combo-1, combo-box using combo-data
        line + 1.5, column 39, lines 5, size 16, 3-d.

    03  push-button, "E&xit Program",
        ok-button, line 25, cline 23, column 27, size 13.


 procedure division.
 main-paragraph.

*    Setup a gray screen background
      display standard graphical window,
      title "PERCobol Components"
      lines 27, size 66, background-low.

*    Add the elements of the table to the combo-box
      modify combo-1, item-to-add = table combo-choice
*    Display the screen and its contents
      display screen-1.

     perform, with test after, until exit-button-pushed

       accept screen-1

*        display this message on the system console
*        so we don't disturb the graphical content.

     display "screen-1 accepted, loop back" upon sysout

    end-perform.

*    Stop All Run is different from Stop Run in that ALL
 *    threads, all graphics are terminated.  In a graphical
 *    program, by default the graphics are left displayed
 *    to prevent the program from being a momentary flash.

     stop all run.
```

Portable Applications

How to take an existing COBOL application written from a where it is to deploy on Linux



(Code Samples)

Just compile is all that should be necessary. If a compile wasn't required, so much the better.

Object Oriented

How to build an OO COBOL program and interchange other objects. A sample of the Object Oriented Cobol directives available in PERCobol.

```
CLASS-ID. name INHERITS "class" IMPLEMENTS "class" ... .
        METHOD-ID. name.
            DATA ...
            PROCEDURE...
        END METHOD.
        METHOD-ID. name.
        END METHOD.
        METHOD-ID. name.
         END METHOD.
        ...
END CLASS.
```

Enterprise Java Beans

How to integrate and/or create Java functionality from COBOL

PERCobol 2.6 supports direct creation of Java classes, with Java capabilities. (This is not yet standard Cobol 2002 standard, but rather close with a Java capabilities. So single-inheritance and multiple implementation is available, exactly as in straight Java.)

```
CLASS-ID. name INHERITS "class" IMPLEMENTS "class" ... .
        METHOD-ID. name.
            DATA ...
            PROCEDURE...
        END METHOD.
        METHOD-ID. name.
        END METHOD.
        METHOD-ID. name.
         END METHOD.
        ...
END CLASS.
```

In PERCobol 2.6, each program and class may be instantiated multiple separate times. (A separate working-storage is created, for example.)

Object references and INVOKE's are also supported. An empty method name in the INVOKE is the constructor.