Jiehong Li
Rona Abraham

COBOL

History of COBOL

- COBOL (**Common Business Oriented Language**) was one of the earliest high-level programming languages.

- COBOL was developed in **1959** by the Conference on Data Systems Languages (CODASYL). This committee was a formed by a joint effort of industry, major universities, and the United States Government. This committee completed the specifications for COBOL as the year of 1959 came to an end. These were then approved by the Executive Committee in January 1960, and sent to the government printing office, which edited and printed these specifications as **Cobol60**. COBOL was developed within a six month period, and yet is still in use over 40 years later.

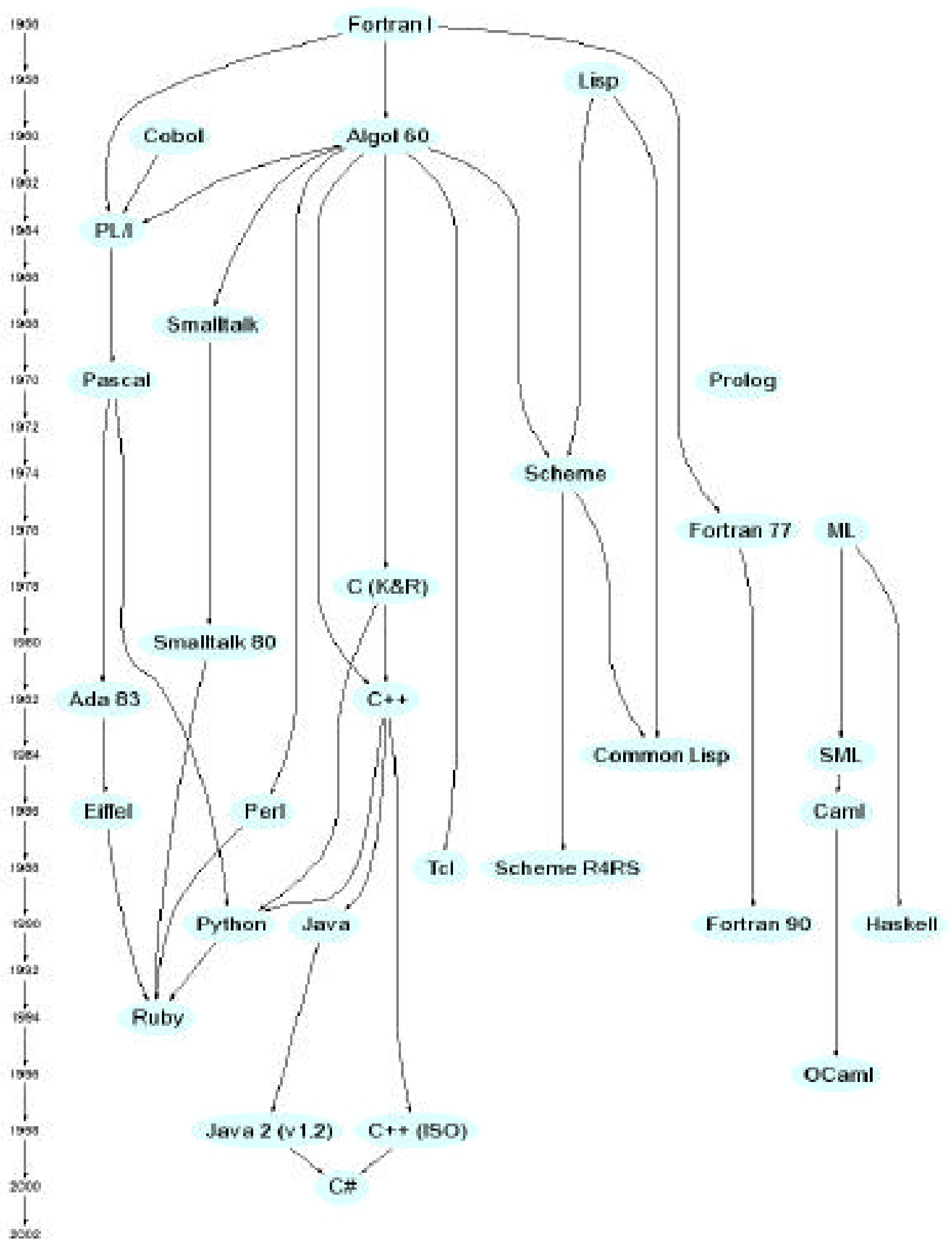
- Since 1960, the American National Standards Institute (ANSI) was responsible for developing new COBOL standards.

- Three ANSI standards for COBOL have been produced: in 1968, 1974 and 1985.

- Object-oriented COBOL is the fourth edition in the continuing evolution of ANSI/ISO standard COBOL.

- The government contributed to COBOL's initial popularity by insisting that computers sold or leased to the government had to have COBOL software available.

Notes:


1956
1958
1960
1962
1964
1966
1968
1970
1972
1974
1976
1978
1980
1982
1984
1986
1988
1990
1992
1994
1996
1998
2000
2002
Fortran I
Lisp
Cobol
Algol 60
PL/I
Smalltalk
Pascal
Prolog
Scheme
Fortran 77
ML
C (K&R)
Smalltalk 80
Ada 83
C++
Common Lisp
SML
Eiffel
Perl
Caml
Tcl
Scheme R4RS
Python
Java
Fortran 90
Haskell
Ruby
OCaml
Java 2 (v1.2)
C++ (ISO)
C#

Notes:

Underlining Philosophy

•Like the name suggests, COBOL was meant to be ‘common’ or compatible among a significant group of manufacturers. i.e. COBOL is non-proprietary .

•COBOL is designed for developing business, typically **file-oriented**, applications, and is not designed for writing systems programs.

Characteristics of COBOL applications

COBOL applications can be very large, with typically more than 1,000,000 lines of code. COBOL applications are also very long-lived. Because of the huge investment in building COBOL applications of a million+ lines, these applications cannot be abandoned whenever a new technology or programming language becomes popular. This is why business applications between 10 and 30 years old are common. This is also why most of the applications affected by the Y2K problem were in COBOL (12,000,000 COBOL applications vs. 375,000 C and C++ applications in the US alone - *Jones, Capers* - *The global economic impact of the year 2000 software problem (Jan, 1997*)

Advantages

-Simple
-Portable
-Maintainable

Disadvantages

-very wordy
-very rigid format
-not designed to handle scientific applications

Notes:

Areas of application

COBOL applications often run in critical areas of business. For instance, over 95% of finance–insurance data is processed with COBOL (*Arranga et al - In COBOL's Defense: Roundtable Discussion (March/April 2000) - IEEE Software*). This is why there was so much panic over the year 2000 problem.

Is COBOL still used?

According to a report from Gartner group,
In 1997 they estimated that there were about 300 billion lines of computer code in use in the world. Of that they estimated that about 80% (240 billion lines) were in COBOL and 20% (60 billion lines) were written in all the other computer languages combined (*Brown, Gary DeWard - COBOL: The failure that wasn't - COBOLReport.com)*

In 1999 they reported that over 50% of all new mission-critical applications were still being done in COBOL and their recent estimates indicate that through 2004-2005 15% of all new applications (5 billion lines) will be developed in COBOL while 80% of **all** deployed applications will include extensions to existing legacy (usually COBOL) programs.

Distinct Features of COBOL

• The language is simple
• No pointers
• No user defined types
• No user defined functions
• ‘Structure like’ data types
• File records are also described with great detail, as are lines to be output to a printer
• COBOL is self documenting

Notes:

Structure of COBOL

COBOL programs are hierarchical in structure. Each element of the hierarchy consists of one or more subordinate elements. The levels of hierarchy are **Divisions, Sections, Paragraphs, Sentences and Statements.** There are 4 main divisions and each division provides an essential part of the information required by the complier. At the top of the COBOL hierarchy are the **four** divisions. The sequence in which they are specified is **fixed**, and must follow the order:

- **IDENTIFICATION DIVISION** supplies information about the program to the programmer and the compiler.
- **ENVIRONMENT DIVISION** is used to describe the environment in which the program will run.
- **DATA DIVISION** provides descriptions of the data-items processed by the program.
- **PROCEDURE DIVISION** contains the code used to manipulate the data described in the DATA DIVISION. It is here that the programmer describes his algorithm.

Some COBOL compilers **require** that all the divisions be present in a program while others **only require** the IDENTIFICATION DIVISION and the PROCEDURE DIVISION.

HelloWorld Example

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. HELLOWORLD.
000300
000400 ENVIRONMENT DIVISION.
000500 CONFIGURATION SECTION.
000600 SOURCE-COMPUTER. RM-COBOL.
000700 OBJECT-COMPUTER. RM-COBOL.
000800
000900 DATA DIVISION.
001000 FILE SECTION.
001100
101200 PROCEDURE DIVISION.
101300
101400 MAIN-LOGIC SECTION.
101500      DISPLAY "Hello world!"
101600 STOP RUN.
```

Notes:

Data Types in COBOL

•COBOL is not a strongly typed language

•In COBOL, there are only three data types
  -numeric
  -alphanumeric (text/string)
  -alphabetic

•Data types are declared using:
  A level number.
  A data-name or identifier.
  A Picture clause.

•e.g.
```
01 GrossPay  PIC 9(5)V99  VALUE  ZEROS.
```

Group Items

•Group items are the COBOL equivalent of structures.

•The items with a group item must be elementary items or other group items.

•Ultimately every group item should be defined in terms of elementary items.

•The hierarchy in a group item is represented by different level numbers

e.g.
```
01 DateOfBirth.
        02  DayOfBirth  PIC 99.
        02  MonthOfBirth PIC 99.
        02  YearOfBirth PIC 9(2).
```

Notes:

Basic Commands in COBOL

ADD *a* TO *b.*
ADD *a* TO *b* GIVING *c.*

SUBTRACT *a* FROM *b.*
SUBTRACT *a* FROM *b* GIVING *c.*

MULTIPLY *a* BY *b.*
MULTIPLY *a* BY *b* GIVING *c.*

DIVIDE *a* INTO *b.*
DIVIDE *a* INTO *b* GIVING *c.*

COMPUTE *x* = *a* + *b* * *c.*

MOVE *a* TO *b* *c.*

SORT *sort-file*
    ON ASCENDING KEY *k*
    USING *inventory-file*
    GIVING *sorted-inventory-file* .

MERGE *merge-work-file*
    ON ASCENDING KEY *K*
    USING *input-file1 input-file2*
    GIVING *output-file* .

DISPLAY *total-cost*.
ACCEPT *identifier.*

PERFORM *paragraphname1* THROUGH paragraphname2
        VARYING *index* FROM *value1* BY *value2*
        UNTIL *condition.*

Notes:

A detailed example in COBOL

```
000010 IDENTIFICATION DIVISION.
```

```
000020 PROGRAM-ID.       SAMPLE.
000030 AUTHOR.           J.P.E. HODGSON.
000040 DATE-WRITTEN.     4 February 2000
000041
000042* A sample program just to show the form.
000043* The program copies its input to the output,
000044* and counts the number of records.
000045* At the end this number is printed.
000046
000050 ENVIRONMENT DIVISION.
000060 INPUT-OUTPUT SECTION.
000070 FILE-CONTROL.
000080     SELECT STUDENT-FILE     ASSIGN TO SYSIN
000090         ORGANIZATION IS LINE SEQUENTIAL.
000100     SELECT PRINT-FILE       ASSIGN TO SYSOUT
000110         ORGANIZATION IS LINE SEQUENTIAL.
000120
000130 DATA DIVISION.
000140 FILE SECTION.
000150 FD  STUDENT-FILE
000160     RECORD CONTAINS 43 CHARACTERS
000170     DATA RECORD IS STUDENT-IN.
000180 01  STUDENT-IN              PIC X(43).
000190
000200 FD  PRINT-FILE
000210     RECORD CONTAINS 80 CHARACTERS
000220     DATA RECORD IS PRINT-LINE.
000230 01  PRINT-LINE              PIC X(80).
000240
000250 WORKING-STORAGE SECTION.
000260 01  DATA-REMAINS-SWITCH     PIC X(2)      VALUE SPACES.
000261 01  RECORDS-WRITTEN         PIC 99.
000270
000280 01  DETAIL-LINE.
000290     05  FILLER              PIC X(7)      VALUE SPACES.
000300     05  RECORD-IMAGE        PIC X(43).
000310     05  FILLER              PIC X(30)     VALUE SPACES.
000311
000312 01  SUMMARY-LINE.
000313     05  FILLER              PIC X(7)      VALUE SPACES.
000314     05  TOTAL-READ          PIC 99.
000315     05  FILLER              PIC X         VALUE SPACE.
000316     05  FILLER              PIC X(17)
000317                 VALUE  'Records were read'.
000318     05  FILLER              PIC X(53)     VALUE SPACES.
000319
000320 PROCEDURE DIVISION.
000321
000330 PREPARE-SENIOR-REPORT.
000340     OPEN INPUT  STUDENT-FILE
000350          OUTPUT PRINT-FILE.
000351     MOVE ZERO TO RECORDS-WRITTEN.
000360     READ STUDENT-FILE
000370         AT END MOVE 'NO' TO DATA-REMAINS-SWITCH
```

```
000380     END-READ.
000390     PERFORM PROCESS-RECORDS
000410         UNTIL DATA-REMAINS-SWITCH = 'NO'.
000411     PERFORM PRINT-SUMMARY.
000420     CLOSE STUDENT-FILE
000430           PRINT-FILE.
000440     STOP RUN.
000450
000460 PROCESS-RECORDS.
000470     MOVE STUDENT-IN TO RECORD-IMAGE.
000480     MOVE DETAIL-LINE TO PRINT-LINE.
000490     WRITE PRINT-LINE.
000500     ADD 1 TO RECORDS-WRITTEN.
000510     READ STUDENT-FILE
000520         AT END MOVE 'NO' TO DATA-REMAINS-SWITCH
000530     END-READ.
000540
000550 PRINT-SUMMARY.
000560     MOVE RECORDS-WRITTEN TO TOTAL-READ.
000570     MOVE SUMMARY-LINE TO PRINT-LINE.
000571     WRITE PRINT-LINE.
000572
000580
```

Notes: