Programming Languages with Hierarchical Structure

Makoto ARISAWA*

Abstract

In designing a programming language for structured programming, a single language seems to be not enough. Since neither "shell approach"(everything in all) nor "core approach"(extensible language) satisfies the present objectives, we propose here a third approach, a language system with hierarchy. ESDL is one such experiment and is discussed briefly. Then we establish desirable relationships between language levels, and discuss what each level should be like. Throughout the paper, the arguments are made on general conceptual basis, and no specific language is proposed.

1. Introduction

There have been several programming languages designed, implemented and used according to their objectives. A single language does not cover all the functions required, nor efficient enough. People tend to write their programs in their favorite languages. We can think of lots of other reasons why we have had many languages so far.

However, there have also been some efforts to cover a wide range of programming by a single language. The expected merits for this are program compatibility, software cost reduction, easier education among other things. These efforts can be classified into shell and core approaches. The shell approach means that one language contains all the functions, where the language complexity is eliminated through modulality and orthogonality. One typical example for this approach is PL/I.

The core approach, on the other hand, means that the language has a fixed set of basic functions, plus a self extensible mechanism. A user extends the language

This paper first appeared in Japanese in Joho－Shori (Journal of the Information Processing Society of Japan), Vol. 17, No. 5 (1976), pp. 370～375.

* Computer Science Department, Yamanashi University

in his own way to fullfil his objectives. One such language example is ELl[13]. These two approaches have their pros and cons. But from the structured programming point of view, neither ones are satisfactory. In this paper, we propose the third approach, language system with hierarchy structure. This is to prepare some compact languages for each level of programming but to keep a sort of uniformity between different levels. ESDL(ETL's System Description Language) is one such example and is described briefly in the next section. The third section discusses structured programming hierarchy.

2. ESDL Hierarchy Structure

ESDL was designed for use of operating systems and other system programming, to cover from general system design through detailed module coding within one language system. ESDL has six different levels named F(highest) through A(lowest) levels, meanings of which are as follows:

F level: flowchart; E level: SIMULA[6]-like simulation language; D level: macro language similar to ALGOL-D[2]; C level: macro base language similar to ALGOL-C[2]; B level: BLISS[16,17]-like high-level assembly language; A level: assembly language. The total structure of ESDL is shown in Fig. 1.

In actual implementation, we found that the gaps between F and E levels, D and C levels, and C and B levels are wider than others, and cannot be transformed automatically. The details of ESDL are found in 10).

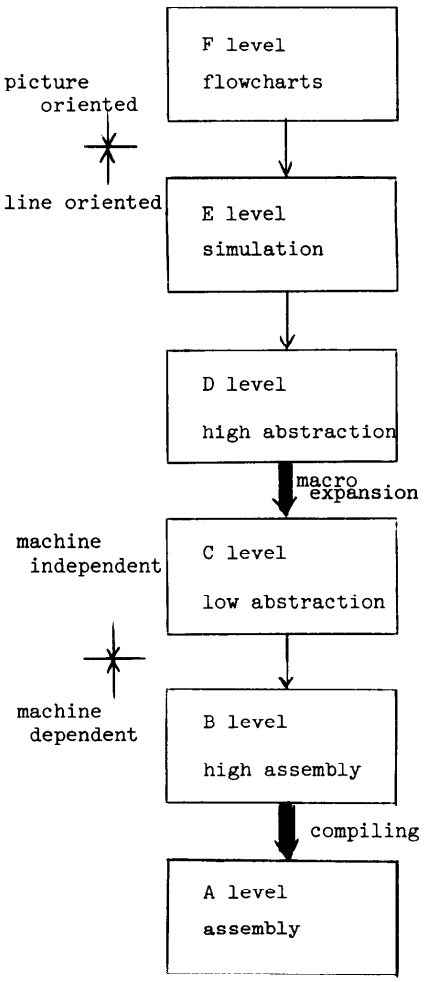

Fig. 1. Six levels in ESDL

3. Hierarchy Structures for Structured Programming

When we mention language levels, we often

mean more than one thing. Here we list exactly what they are.

(1) Programs in the higher level are also accepted as programs in the lower level (but not vice versa).

(2) Programs in the lower level are also accepted as programs in the higher level (but not vice versa).

(3) Control structures allowed in the higher level are also allowed in the lower level.

(4) Data structures and operations allowed in the higher level are also allowed in the lower level.

(5) Control structures in the lower level are refinements of the ones in the higher level.

(6) Data structures and operations in the lower level are refinements of the ones in the higher level.

(7) Language processor for the higher level can be implemented in the lower level.

(8) Programs in the higher level can be transformed into the lower level through macro expansions.

(9) Programs in the higher level can be transformed into the lower level through language processors.

Examples of above relationships are as follows. (1): machine independent macro assembly language (high) vs machine dependent macro assembly language (low); (2): macro assembly language (high) vs assembly language without macros (low); (5): if-then-else and while-do constructs being implemented in branch instructions; (6): Matrix structure and operations being accomplished in arrays and repetition of scalar operations; (7) ALGOL-W compiler being implemented in PL360. Other relations will be obvious.

Now we consider each language level from the structured programming point of view.

(i) F level (highest) is for description of the problem recognition, and algorithms to solve it are roughly described. This level corresponds to HIPO[5] and structured design[12].

(ii) E level is for conceptual description of algorithms, allowing any control structures or data structures/operations. Class concept in SIMULA67 should be incorporated in this level.

(iii) D level is for refined algorithm description, and control structure should

be restricted within a fixed well-formed set.

(iv) C level is for detailed algorithm description, and the data structures/ operations are also restricted to allow a programmer to consider the efficiency of the algorithm implementation. This efficiency is bought by losing some sort of dynamic freedom, and PASCAL[3,15] will be an good example for finding trade-offs in generality vs efficiency balance.

(v) B level is for machine dependent description of the algorithms, and the specific machine features can be taken into the programs at this level. BLISS gives one typical B level idea, but the accessing path concept in BLISS might not be of use for IBM/360-370 -like computers and PL360 indicates an alternative.

(vi) A level is macro assembly language, and is often supplied by the manufacturers.

Desirable between-level relationships (1) through (6) are shown in Fig. 2. As to relations (7), (8) and (9), the actual issue is to consider in which level a user wants transformation done automatically and where he likes to do it by hand. With structured programming, important decisions must be made in higher levels. Those decisions that can be postponed until a later levels are bearable for heavy descriptive restrictions. Therefore, lower levels should be designed in such a way that automatic transformations are not of much difficulty, while higher levels should allow lots of freedoms for users. In other words, generality for higher levels and efficiency for lower levels, as our common sense might indicate.

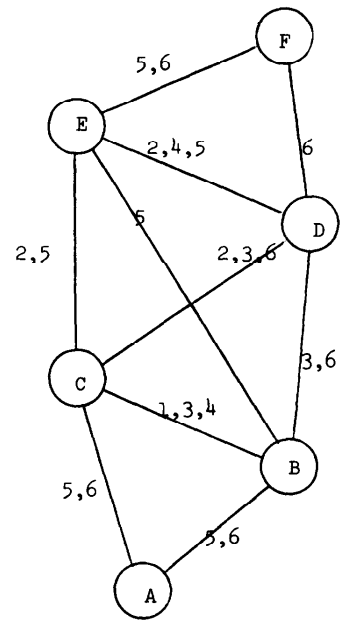

Fig. 2. Relations between the levels

4. Conclusion

We proposed six-level language structure, but did not mention any specific language specifications. This is because general design principles should be discussed separately from a specific programming language design as in Hoare[4]. When an

actual language system is to be designed, the most important thing is to set the between-level gap properly, based on the level-to-level balance. The number of levels, six, is not of much importance but is only taken from ESDL for comparison.

References

1) O.J.Dahl, E.W.Dijkstra, C.A.R.Hoare: Structured programming. Academic Press (1972)

2) B.Galler, A.J.Perlis: A proposal for definition in ALGOL. CACM 10-4,204-219 (1967)

3) A.N.Habermann: Critical comments on the programming language PASCAL. Acta Informatica 3,47-57 (1973)

4) C.A.R.Hoare: Hints on programming language design. STAN-CS-73-403 (1973)

5) IBM: Management overview. Internal report (1973)

6) J.D.Ichibiah, S.P.Morse: General concepts of the SIMULA 67 programming language. Annual Review in Automatic Programming 7,65-93 (1973)

7) D.E.Knuth: Structured programming with go to statements. Computing Surveys 6-4,261-301 (1974)

8) D.E.Knuth: A review of structured programming. STAN-CS-73-371 (1973)

9) C.Lang: SAL - system assembly languages. SJCC 69,543-555 (1969)

10) N.Saito, Y.Suguito, M.Arisawa, M.Miyakawa, M.Sato: ESDL. Bul.ETL 34-5/6 (1970)

11) R.L.Sites: ALGOL-W reference manual (revised). STAN-CS-71-230 (1972)

12) W.P.Stevens, G.J.Myers, L.L.Constantine: Structured design. IBM Systems Journal 74-2,115-139 (1974)

13) B.Wegbreit: The ECL programming system. FJCC 71,253-262 (1971)

14) N.Wirth: PL360 - a programming language for the 360 computers. JACM 15-1,37-74 (1968)

15) N.Wirth: The programming language PASCAL. Acta Informatica 1,35-63 (1971)

16) W.A.Wulf: Programming without the go to. IFIP 71,408-413 (1972)

17) W.A.Wulf, D.B.Russell, A.N.Habermann: BLISS - a language for system programming. CACM 14-12,780-790 (1971)