Adaptable Systems Linguistic and Processing Properties. Part Two.

Dumitru TODOROI, Diana MICUŞA, Zinaida TODOROI
todoroi@ase.md, ASE Chişinău

As continuation of the research results [Tod-05.3] in this paper metalinguistic tools used in the process of introduction of new constructions (data,, operations, instructions and controls) are developed. It was done the overview of extensible languages according used extension mechanisms, presented by different types of extenders. Extensions in languages and systems are defined according different phases of translation such as lexical analysis, syntactical control, semantic analysis, translation into intermediate language, and link phase. It was presented a lot of extensible languages and systems and examples of usage of different types of extenders. The generalization schemes of evaluation of adaptable languages and systems are discussed. Development of the research in Adaptable Programming Basis [Tod-05.1,2,3] is presented. These results analogically with [Tod-05.2,3] are obtained by the team, composed from the researchers D. Todoroi [Tod-05.4], Z. Todoroi [ZTod-05], and D. Micusa [Mic-03]. Present results are included in the book [Tod-06].
Keywords: *processor, adaptable systems, linguistic processing.*

Introducere

Societatea informațională, bazată pe integritatea de calculatoare, interfețe şi rețele, prin intermediul interacțiunilor adaptabile (simple şi naturale) va face accesibile multiplele servicii şi aplicații în viața de toate zilele a societății umane. Această viziune a mediului intelectual adaptiv interactiv cere instalarea utilizatorului şi a bunăstării populației în centrul viitoarei societăți, bazate pe cunoştințe, în evoluarea logică a societății [Tod-05.1, Mic-04] de la **societatea informațională** la **societatea cunoaşterii** şi în continuare la **societatea conştiinței.**

Extensibilitatea şi, în dezvoltare, ***adaptabilitatea*** ca idee în tehnologiile informaționale [Tod-05.1,2,3] şi, în particular, în limbajele şi sistemele de programare au evaluat odată cu evoluțiile calculatoarelor de la generație I-a până azi – la generația a V-a.. Primele limbaje şi sisteme orientate la calculator (I-a generație) susțineau deja idei de extindere de comenzi, instrucțiuni şi ***subprograme,*** utilizate prin intermediul ***definițiilor şi apelurilor***. În sistemele orientate la specialist sau la un grup de specialişti (generația a II-a şi a III-a) extensibilitatea este deja prezentată prin ***macrouri, subrutine, structuri şi tipuri*** cu aceeaşi modalitate de ***definire şi apel.*** Sistemele orientate la obiect, acțiuni, probleme şi apoi la utilizatorii profesionalişti şi neprofesionişti (generația a IV-a şi a V-a) sunt caracterizate printr-un set de ***facilități extensibilo-adaptabile***, care pot fi divizate în facilități lingvistice şi cele de sistem.

Primele două forumuri internaționale, consacrate în special limbajelor şi sistemelor extensibile [Pro-69,71] au evidențiat proprietățile şi facilitățile limbajelor şi sistemelor extensibile şi au accentuat problemele, care apar în această direcție de cercetări şi implementări a sistemelor de interacțiune om-maşină. Prima generalizare a rezultatelor, obținute în domeniul extensibilității, a fost efectuată de Solntseff şi Yezerski [Sol-74] din punct de vedere al divizării mijloacelor extensibile în dependență de fazele de translatare. În țările FUS şi în continuare CSI, în anii 1982-1993 a activat o grupă specială de interese "Sisteme adaptabile de programare" [Cuz-89, Tod-87], care la întrunirile sale anuale analiza starea de evaluare a problemelor extensibilității şi adaptabilității din punct de vedere lingvistic, sistemic şi realizatoric - pragmatic [Tod-92]. Ultimii 20 ani la forumurile anuale internaționale OOPSLA-1987 - OOPSLA-2006 problemele adaptabilității sunt intensiv abordate ca mijloace adaptabile a programării obiect – orientată şi acțiune-orientată. Realizările RAB, IADRO,

SAGRED3D, C++, Ada şi Java sunt confirmări practice ale teoriei şi practicii programării adaptabile.
În această lucrare – continuare a lucrărilor [Tod-05.2,3] vor fi analizate şi dezvoltate problemele adaptabilităţii limbajelor prin definiţii de extensii de limbaj. Sunt prezentate extensii de date, operaţii, instrucţiuni şi dirijări. Sunt enumărate un set de proiecte de sisteme extensibile şi adaptabile. Sunt prezentate rezultatele de generalizare a noţiunii adaptabilităţii începând cu Baza adaptabilă şi continuând cu nivelurile limbajelor adaptabile. Sunt analizate metodele de evaluare a limbajelor şi sistemelor adaptabile: metodele lingvistice BOTTOM-UP, TOP-DOWN şi HORIZONTAL, de asemenea şi metodele evaluate de creare a limbajelor adaptabile şi a sistemelor adaptabile.

1. Adaptabilitate şi extensibilitate.
Cum este evidenţiat în [Tod-05.2], mijloacele metalingvistice, destinate introducerii de noi construcţii (date, operaţii, instrucţiuni, dirijări) în limbaj se numesc **extensii de limbaj.**
Aceste extensii sunt elemente de limbaj şi sunt părţi componente a limbajului Baza adaptabilă. Extensia de limbaj este realizată prin intermediul **definitorului** şi **apelului extensiei**.
Definitorul de extensie permite extinderea limbajului cu noi elemente derivate ale limbajului. El realizează definirea extensiei.
Definitorul de extensie la fel ca şi definitorul de reducere (individualizare) includ:
- definirea pragmaticii elementului-extensie,
- definirea sintaxei elementului-extensie,
- definirea semanticii elementului-extensie,
- definirea contextului de utilizare a elementului-extensie,
- prezentarea exemplelor de apel a elementului-extensie.

Adaptorul este compus din definitor de extindere (**Adaptor-extendor**) şi definitor de reducere (**Adaptor-reductor**).
Schema generală a **adaptorului-extendor** este următoarea:
_BL_ pragmatica elementului-extensie
_SY_ sintaxa elementului-extensie
_SE_ semantica elementului-extensie
_CO_ contextul de utilizare a elementului
_EX_ exemple de apel a elementului-extensie
_EL_
Din definiţie reiese, că părţile componente ale definitorului elementului-extensie [Tod-05.2], sunt pragmatica, sintaxa, semantica, mediul de utilizare si exemplele de apel a noului element definit.
Pragmatica identifică locul, necesitatea extensiei în limbaj, explică necesitatea includerii în limbaj a elementului nou. **Sintaxa** defineşte forma de apel a extensiei –elementului nou. **Semantica** reprezintă descrierea conţinutului elementului nou prin intermediul conţinutului altor elemente deja introduse în limbaj. **Contextul** apreciază locul de utilizare a acestor elemente derivate, iererhia lor în cadrul elementelor limbajului. **Exemplele** conţin exemple de apel şi de utilizare a elementului nou introdus.

2. Extensii de limbaj.
Utilizarea **adaptorului-extendor** este efectuat prin intermediul utilizării modificărilor lui la prezentarea exemplelor de extensii de date, operaţii, instrucţiuni şi dirijări. Prezentarea fiecărei din extensii este precedată de prezentarea mediului operaţional, în care această extensie va avea loc. Extensia este prezentată prin intermediul schemei generale a **adaptorului-extendor** evaluată pentru cazurele de prezentare a extensiilor de date, operaţii, insrucţiuni şi dirijări.
2.1. Extensii de date.
Dorim să declarăm o dată nouă cu numele de <vector>. Această dată va fi exprimată semantic printr-un element de nivel inferior cu numele de <ARRAY>.
Mediul operaţional de definire a elementului-extensie-dată **<vector>.** Sunt cunoscute:
(1) noţiunile neterminale: <type>, <identif>, <bj> (bariera de jos), <bs> (bariera de sus),
(2) simbolurile metalingvistice: ¯DBL¯ , ¯DSY¯ , ¯DES¯ , ¯DEX¯, ¯DEL¯ şi ¯AS¯ şi
(3) elementele terminale B, C, 10, 1, 200, ARRAY, [,], -, :, < şi >.
În acest mediu noul elementul-extensie de date **<vector>** va fi definit prin intermediul **adaptorului-extendor de date** în forma ur-

mătoare:

-DBL- <vector>
-DCO- -AS- <type>
-DSY- <identif> : VECTOR [<bj>:<bs>]
-DSE- ARRAY <identif> [<bj>:<bs>]
-DEX- B:VECTOR [-10:10]
C: VECTOR [1:200]
-DEL-

Acest element este de nivelul "i+1" şi este definit prin intermediul elementelor nivelurilor de mai jos, inclusiv a elementelor de nivel "i": ARRAY, <identif>, <bj>, <bs>, şi altele. Mediul (Contextul) de utilizare a acestui element <vector> **de tip date** coincide cu mediul de utilizare a oricărei alte date de tip <type>.

2.2. Extensii de operaţii

De definit o nouă operaţie cu numele de <sum vector>: sumare a vectorilor.

Mediul operaţional de **definire** a elementului-extensie-operaţie **<sum vector>.** Sunt cunoscute:

(1) operaţia "+" (sumare),
(2) noţiunile de identificatori: <identif1> şi <identif2>,
(3) instrucţiunile de atribuire, înscriere a rezultatului şi cele de dirijare a ciclului. Sunt cunoscute sistemului adaptabil de asemenea:
(4) funcţia **Length**,
(5) metaterminalele: -OBL-, -OCO-, -AS-, -OSY-, -OSE-, -OEX- şi -OEL- şi
(6) terminalele: A, M, 1, k, x, y, P, Q, VECTOR, CYCLE, FROM, TO, BYSTEP, DO, ENDCYCLE, RESULT, <, >, [,], (,), +,:, ;, := şi =.

În acest mediu noul elementul-extensie de operaţii **<sum vector>** va fi definit prin intermediul **adaptorului-extendor de operaţii** în forma următoare:

-OBL- <sum vector>
-OCO- -AS- <sumare>
-OSY- <identif1> + <identif2>
-OSE- A := Length (<identif1>);
k : VECTOR [1:A];
CYCLE M FROM 1 TO A BYSTEP 1 DO
k[M]:= <identif1> [M]+ <identif2 > [M];
ENDCYCLE;
RESULT k: VECTOR;
-OEX- x + y

P + Q

-OEL-

Pentru această definiţie se presupune, că numărul elementelor vectorilor cu numele de <identif1> şi de <identif2 > sunt egale.

2.3. Extensii de instrucţiuni

De definit o nouă instrucţiune cu numele de <Lin-Frântă>.

Mediul operaţional de **definire** a elementului-extensie-instrucţiune **<Lin-Frântă>.** Sunt cunoscute:

(1) datele de tip ARRAY,
(2) instrucţiunile <ciclu>, <punct>, şi <declarare>,
(3) funcţia **Length** şi tipul **INTEGER**,
(4) metadelimitatorii: -SBL-, -SCO-, -AS-, -SSY-, -SSE-, -SEX-, -SEL- şi -KIND- şi
(5) elementele terminale: POINT, CYCLE, FROM, TO, BYSTEP, DO, SEGMENT, END_CYCLE, [,], (,), <, >, :, ;, x, y, M, 1, 2, 3, T, S, ARRAY, k, kx, ky, kTX, kTY, P.

În acest mediu noul element-extensie de instrucţiune **<Lin-Frântă>** va fi definit prin intermediul **adaptorului-extendor de instrucţiuni** în forma următoare:

-SBL- <Lin-Frântă>
-SCO- -AS- <punct>, <ciclu>,
-SSY- -KIND-:1: BROKEN_LINE (x, y);
-KIND-:2: BROKEN_LINE (x: ARRAY [1:H], y: ARRAY [1:H]);
-KIND-:3: BROKEN_LINE (x,y: ARRAY; H: INTEGER);
-SSE- -KIND-:1: H:= Length (x);
-KIND-:1,2,3: POINT (x[1], y[1]);
CYCLE k FROM 2 TO H BY STEP 1 DO
SEGMENT (x[k], y[k]);
END_CYCLE

```
-SEX-   BROKEN_LINE (T,S);
        BROKEN_LINE (kx: ARRAY [1:k], ky:ARRAY [1:k])
        BROKEN_LINE  (kTX, kTY: ARRAY; P:INTEGER)
-SEL-
```

Pentru această definiție se presupune, că numărul elementelor din perechile de tablouri-vectori cu numele de x şi y, T şi S, şi kTX şi kTY sunt egale.

2.4. Extensii de dirijări

Presupunem, că este necesar de definit o nouă dirijare cu numele de <Cycle>, care la acest moment nu este încă inclus în sistemul adaptabil.

Mediul operațional de **definire** a elementului-extensie-dirijare **<Cycle>.** Sunt constituie următoarele elemente:

(1) funcția SIGN, expresia <exp2> şi instrucțiunea <statement>,
(2) expresiile <ep>, <ae1>, <ae2>, <ae3>,
(3) instrucțiunile <compusă> şi <atribuire>,
(4) dirijările <conditional> şi <transfer>,
(5) terminalele: [,], (,), <,>, +, -, *, :, :=, ≤, 1, 2, H, K, P, M1, M2, S, A, I, C, D, I, SIGH, WHILE, DO, FOR, TO, BY_STEP, IF, THEN, GOTO, BEGIN şi END şi
(6) metaterminalele: : -CBL-, -CCO-, -AS-, -CSY-, -CSE-, -CEX-, -CEL- şi -KIND-.

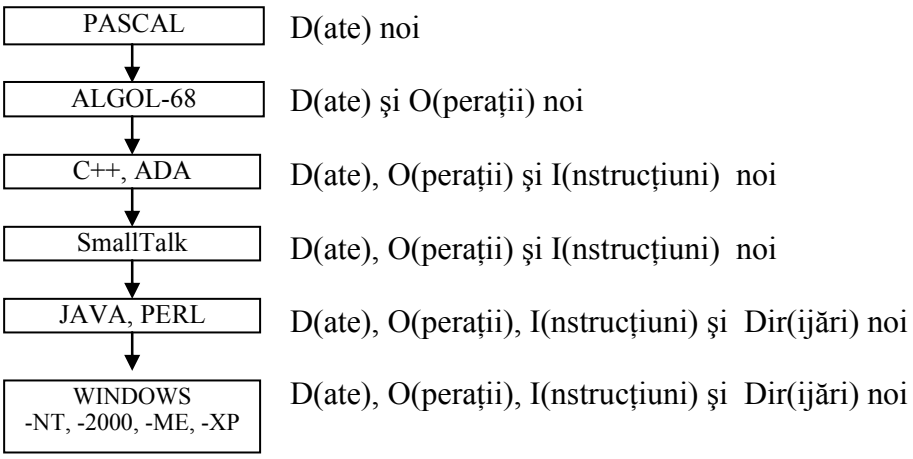
În acest mediu noul elementul-extensie de instrucțiune **<Cycle>** va fi definit prin intermediul **adaptorului-extendor de dirijări** în forma următoare:

```
-CBL- <cycle>
-CCO- -AS- <statement>
-CSY-  -KIND-:1: WHILE <exp2> DO  <statement>;
       -KIND- :2: FOR <ep>:= <ae1>  TO  <ae2> BY_STEP <ae3> DO <statement>;
-CSE-                                                  -KIND-          :1:
            M1: IF  <exp2> THEN BEGIN <statement>; GOTO M1 END; GOTO M3;
       -KIND-                                                          :2:
            <ep>:= <ae1>;
            M2:  IF  (<ep>-<ae2>)*  SIGN  (<ae2>)≤0  THEN
            BEGIN  <statement>;  <ep>:=<ep>  +  <ae3>;GOTO  M2  END;
            M3: ;
-CEX-  WHILE  H<K DO BEGIN S:= S+A[H]; H:=H+P; END;
       FOR I:=1 TO H  BY_STEP 2 DO BEGIN C:= C + A[I]; D:= D + A [I+1] END;
-CEL-
```

A fost definită o nouă extensie-dirijare cu numele de <Cycle>. Ea este definită în două forme: WHILE-DO şi FOR-DO. Semantica acestor doua elemente-dirijări este definită prin intermediul elementelor-dirijări de un nivel mai jos : IF-THEN şi GOTO, instrucțiunea de atribuire şi instrucțiunea compusă BEGIN-END.

3. Evaluarea limbajelor si sistemelor adaptabile

Extendorii şi în continuare adaptorii [Tod-04, Mic-04, Tod-92] formează nucleul de creare şi realizare a unei serii de proiecte şi sisteme adaptabile şi extensibile reale şi/sau experimentale. Aceste limbaje şi sisteme extensibile-adaptabile se caracteriuează prin extinderea seturilor lor de D(ate), şi/sau O(perații), şi/sau I(nstrucțiuni), şi/sau de Dir(ijări). Cele mai reprezentative limbaje şi sisteme extensibile-adaptabile in evoluția linbajelor şi sistemelor de programare sunt: PASCAL, ALGOL-68, C++, ADA, SmallTalk, JAVA, PERL, WINDOWS-NT, WINDOWS-2000, WINDOWS-ME, WINDOWS-XP.



3.1. **Dirijările interne** sunt obținute [Tod-05.2] prin noi facilități de dirijare în cadrul unui program. Acestea sunt dirijările susținute de ierarhia operațiilor la executarea unei expresii.

Dirijările instrucțiunilor sunt efectuate de instrucțiunile-dirijări de tipul ramificări, cicluri, adresări la subrutine, proceduri etc.

3.2. **Dirijările externe** între diferite programe şi între maşini le constituie metodele de multiprogramare şi multiprocesare, executarea virtuală a programelor, sistemele de divizare a timpului etc.

Exemple de dirijări externe (multiprocesare): Maşini vectoriale, Maşini recursive, Rețele Bus, Rețele Ring, Rețele Star

Un exemplu clasic de ierarhizare a unui sistem îl constituie divizarea elementelor limbajului ALGOL-60, care au fost împărțite în 17 niveluri [Tod-68.1,2]. Translatorul recursiv – modular [Tod-68.1] este unul din primele translatoare din lume, care a realizat limbajul ALGOL – 60 la maşini de generația I-a şi a II-a. El a fost pe deplin modular şi suficient de efectiv din punct de vedere a timpului de execuție a programelor ALGOL şi şi eficient din punct de vedere a memoriei ocupate în comun de translator şi programul-obiect obținut de translator. Motorul acestui translator l-a constituit „Sistemul de implementare a programelor recursive: SIRP" [Tod-67], care a fost partea componenta a translatorului ALGOL-60.

Un şir de sisteme adaptabile, extensibile, reductibile au fost realizate şi de grupul de cercetători din Republica Moldova [Tod-80,83,87,92] într-o perioadă îndelungată, începând cu anii şaptezeci până în prezent cu concursul metodelor evoluate de interacțiuni translatorice. Proiectele realizate de autorii acestui grup de cercetători în baza tehnologiei creării, implementării şi evaluării sistemelor modulare şi recursive [Tod-68.1, 80, 83, 87, 89], dezvoltate de ei la fel ca şi a celor extensibile şi adaptabile [Tod-92, 98, Tod-04], sunt următoarele:

- Translatorul recursive "TIGRA" (anul de dare în exploatare: 1968)
- Sistema de grafică pe calculator "GRAFIC" (anul de dare în exploatare: 1975)
- Sistema extensibilă de grafică pe calculator "G1" (anul de dare în exploatare: 1976)
- Sistema universală extensibilă "MACROSYSTEMA-77" (anul de dare în exploatare: 1977)
- Baza algoritmică extensibilă "RAB" (anul de dare în exploatare: 1980)
- Sistema extensibilă de grafică pe calculator "YADRO" (anul de dare în exploatare: 1985)
- Sistema adaptabilă universală "COMSYBASIC" (anul de dare în exploatare: 1988)
- Proiectul "PRARM" (anul de dare în evaluare: 1990)
- Sistema extensibilă de grafică pe calculator "CRED3D" (anul de dare în exploatare: 1997)
- Sistema adaptabilă universală "SAGRED3D"(anul de dare în exploatare: 2003)

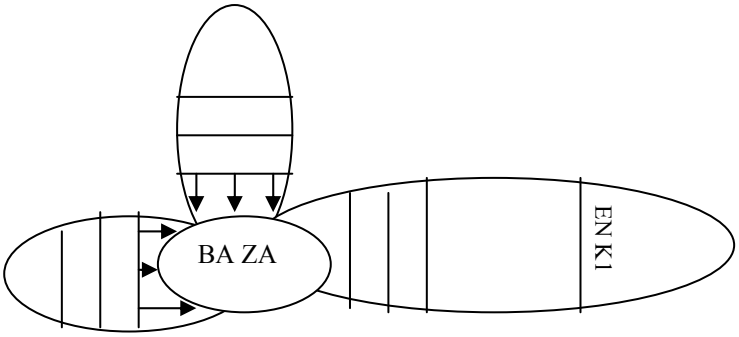
Multe sisteme adaptabile (extensibile şi reductibile) au fost realizate, începând cu anii

şaptezeci [Ear-70, Sol-74] şi până în prezent [Tod-92, Tod-04] cu concursul multor metode variate de interacţiuni translatorice [Mic-04,03].

O bibliografie imensă de idei şi realizări de limbaje şi sisteme orientate-obiect, orientate-agent, extensibile şi adaptabile, care sunt bazate pe ideea extensibilităţii şi adaptabilităţii, poate fi selectată doar enumerând comunicările, publicate in cadrul ultimelor 20 de ani (1987-2006) de conferinţe anuale internaţionale OOPSLA.

4. Generalizări

Adaptorul într-un sistem adaptabil este compus din **extendor** si **reductor**. El activează pentru a extinde şi/sau reduce limbajul adaptabil şi translatorul adaptabil corespunzător. Schema generala a sistemului adaptabil este:



Baza este compusă din extendori, reductori şi elementele de bază, incluse axiomatic în limbajul adaptabil şi sistemul respectiv. Elementele de nivel sunt definite prin intermediul extendorilor şi reductorilor.

Prin intermediul **extendorului** (definire şi apel) se obţine extinderea limbajului şi a sistemului adaptabil. Această operaţie poate fi efectuată prin intermediul extinderii de un anumit tip: date, instrucţiune, operaţie sau dirijare. Prin extinderi succesive se ajunge de la bază la nivelul limbajului adaptabil necesar EnKi, scopul final fiind obţinerea limbajului natural de comunicare formalizat.

Prin intermediul **reductorului** se obţine un sistem de extinderi individuale. În acest caz utilizatorul evidenţiază elementele necesare pentru sistemul adaptabil de programare individual, tipul procesoarelor, care vor activa în acest sistem individual şi formele de trecere dintr-un nivel in altul. Rezultatul acestei reduceri îl constituie sistemul adaptabil individual. Acest sistem va fi compus din baza extensibilă şi elemente de nivel deja definite în acest sistem adaptabil.

Se pot efectua extinderi şi reduceri în diferite direcţii. Sistemul adaptabil este un sistem universal şi specializat. El dispune de multe alte proprietăţi [Tod-05.2,3].

4.1. Baza şi nivelurile limbajului adaptabil

Fie dat $T = \{t_1, t_2, \dots, t_n\}$ – un alfabet al terminalilor limbajului adaptabil de programare. Limbajele adaptabile au o structură ierarhică [Tod-06]. Elementul cu numărul "i" a nivelului "j" se notează prin E_iL_j. Elementul E_iL_j este definit ca o functie:

$E_iL_j = \varphi_{ij} (T, EL_0, EL_1, \dots, EL_{j-1}, EL_j', Elj'')$, unde:

(1) $j = 1, 2, \dots, P$,

(2) P - numărul de niveluri ale limbajului adaptabil,

(3) $i = 1, 2, \dots, N_j$,

(4) N_j – numarul de elemente ale nivelului "j",

(5) $ELk = (E_1Lk, E_2Lk, \dots, E_{Nk}Lk)$ – setul de elemente ale nivelului "k" şi

(6) $k = 0, 1, \dots, j$ – numărul nivelului limbajului adaptabil, elementele căruia se definesc.

În procesul de definire a elementelor este necesar de evidenţiat **situaţiile** suplimentare:

(1.a) Setul de elemente $EL_j' = (E_1L_j', E_2L_j', \dots, E_{Nj}'L_j')$ este compus din elementele nivelului "j", prin intermediul cărora este definit elementul E_iL_j al acestui nivel "j" inclusiv şi printr-o recursie directă sau indirectă.

(1.b) Setul de elemente $Elj'' = (E_1Lj'', \dots, EN_{Nj}''Lj'')$ reprezintă mulţimea elementelor nivelurilor superioare elementului E_iL_j, prin intermediul cărora sunt definite elementele acestui nivel "j" şi printr-o posibilă recursie

indirectă.

(1.c) Funcția $\varphi_{i,j}$ este funcția de realizare a elementului "i" de pe nivelul "j" cu ajutorul a cel putin a unui element de pe nivelul "j-1", unde i = 1,2,…, N_j şi j = 1,2,…,P;

(1.d) BAZA adaptabilă: elementele E_iL_O din EL_O, unde i = 1,2,…, N_0 şi N_0 - numărul elementelor bazei limbajului adaptabil, sunt definite axiomatic.

Exemplu: Expresia aritmetică din limbajele C++, ADA, ALGOL-68, PERL, de exemplu, este definită prin intermediul producțiilor următoare:

< expresia aritmetică primară> ::= <constantă> | <variabilă> | <apel de funcție> | (<expresia aritmetică >) : **situația (1.b): recursie indirectă**

<factor> ::= < expresia aritmetică primară> | <factor> ↑ < expresia aritmetică primară> : **situația (1.a): recursie directă**

<term> ::= <factor>| <term> * <factor> : **situația (1.a): recursie directă**

< expresia aritmetică simplă> ::= <term>| < expresia aritmetică simplă> + <term> : **situația (1.a): recursie directă**

<expresia aritmetică>::= < expresia aritmetică simplă> | < expresia aritmetică condițională>

4.2. Metodele adaptabile de evaluare procesorală

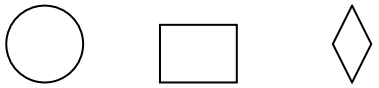
Există trei tipuri de creştere a limbajelor şi sistemelor adaptabile:

- creştere în sus (spre utilizator) – metoda adaptabilă procesorală BOTTOM-UP,
- creştere în jos (spre Bază) – metoda adaptabilă procesorală TOP-DOWN,
- creştere orizontală – metoda adaptabilă procesorală HORIZONTAL.

4.2.1. Metoda adaptabilă procesorală BOTTOM-UP. Creşterea în sus a limbajului adaptabil se efectuează cu ajutorul extendorilor, prin intermediul cărora elementul nivelului "i" participă împreună cu celelalte elemente de nivel mai jos la definirea elementului-extensie, care aparține nivelului "i+1". Acest proces poate fi efectuat în momentele creşterii limbajului pâna la obținerea sistemului adaptabil, orientat al utilizatorului. Acest proces se numeşte **creşterea BOTTOM-UP** a limbajului adaptabil.

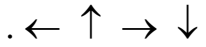
4.2.2. Metoda adaptabilă procesorală TOP-DOWN. Creşterea în jos a limbajului adaptabil de asemenea este efectuiază cu ajutorul extendorilor. Efectul acestei evaluări constă în aceia, că elementele de bază sunt extinse spre diferite dispozitive ale calculatorului. Acest proces se numeşte **creşterea TOP-DOWN** a limbajului adaptabil.

Exemplu, fie dat un set de dispozitive grafice cu diferite niveluri ale limbajelor de programare din grafica pe calculator: analitică şi reprezentativă. Unele grafplotere, ca dispozitive anexate la calculator, pot doar să apropie creionul de hârtie, să-l ridice şi să-l mişte în dreapta, stânga, sus, jos (Nivelul cel mai jos de comunicare). Alte dispozitive pot trasa linii orizontale, verticale, oblice, curbe (Nivel intermediar). Următorul dispozitiv de nivel mai înalt de posibilități (Nivel apropiat de baza limbajului adaptabil) poate trasa diferite figuri. In ordinea discreşterii "intelectualității" dispozitivelor, ele au următoarele posibilități:



Nivel apropiat de baza grafică a limbajului adaptabil

Nivel intermediar



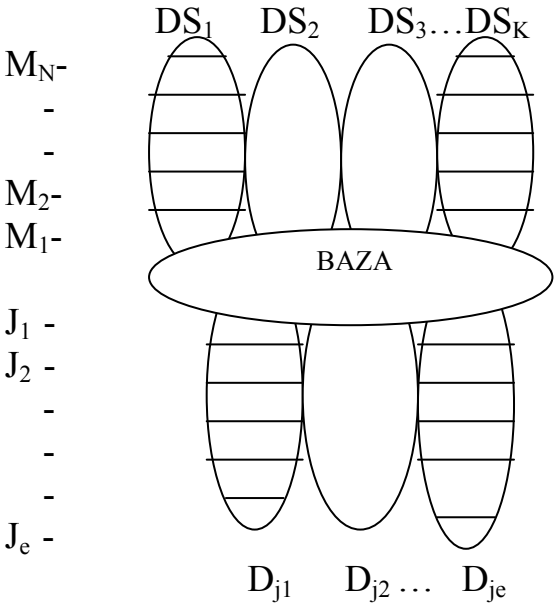
Nivelul cel mai jos de comunicare grafică

Evaluarea în jos permite utilizarea elementelor de cel mai înalt nivel, dar care imediat urmează nivelul elementelor Bazei limbajului adaptabil. Aceste elemente cu ajutorul extendorilor se realizează prin intermediul elementelor următoarelor niveluri de mai jos (De exemplu: nivelul intermediar). Aceste elemente de nivel mai jos se realizează prin intermediul elementelor mai aproape de nivelul de comunicare cu grafploterul cu un „inte-

lect" foarte redus (nivelul cel mai jos de comunicare grafică).

4.2.3. Metoda adaptabilă procesorală HORIZONTAL. Prin creşterea orizontală a limbajului adaptabil se subînțelege, că elementele bazei limbajului adaptabil împreună cu elementele altor niveluri de sus şi de jos pot fi extinse prin adăugarea altor elemente (extindere directă sau extindere recursivă).

În aşa fel se obțin dialectele de sus ale limbajului adaptabil, compuse din elemente derivate după metoda Bottom-Up de creştere a limbajului: DS1, DS2, ... , DSK şi dialectele de jos ale limbajului adaptabil, compuse din elementele derivate după metoda Top-Down: D_{j1}, D_{j2} , … , D_{je} . Extinderea orizontală a limbajului adaptabil se efectuează după metoda Horizontal si prin intermediul ei se îmbogățesc atât nivelurile dialectelor de sus şi de jos ale limbajului adaptabil cât şi baza. Schema limbajului adaptabil, în aşa fel este următoarea:



4.3. Metodele de evaluare a limbajelor adaptabile

Metodele lingvistice de adaptare procesorală depind de următoarele situații lingvistice, în care activează utilizatorul-programator, susținut de sistemul adaptabil.

Situația (3.a): Prin intermediul elementelor lingvistice ale sistemului adaptabil pot fi realizate problemele utilizatorului. **Soluția:** Limbajul adaptabil nu se modifică.

Situația (3.b): Problema nouă cere adăugarea unor elemente de nivel superior celor existente din limbajul adaptabil. Aici P este numărul prezent de niveluri ale limbajului adaptabil.

Soluția:: Limbajul adaptabil se modifică prin adaosul elementelor nuvelurilor superioare celor existente în limbaj: $\mathbf{EL_{p+1}, EL_{p+2}, \ldots, EL_{p+q}}$

Observație: Extinderea limbajului adaptabil este reprezentat prin definirea nivelurilor noi ale limbajului (Modelul Bottom-Up).

Situația (3.c): Problema cere adăugarea unor elemente noi la nivelurile deja existente ale limbajului adaptabil.

Soluția: Limbajul adaptabil se modifică prin adaosul elementelor noi la nivelurile existente în limbaj:

$EL_o = \{E_1L_o , \ldots, E_{No}L_o\} \cup \mathbf{\{E_{No+1}L_o , E_{No+2}L_o, \ldots, E_{No+Po}L_o\}}$

$EL_1 = \{E_1L_1, \ldots, E_{N1}L_1\} \cup \mathbf{\{E_{N1+1}L_1 , E_{N1+2}L_1, \ldots, E_{N1+P1}L_1\}}$

.....

$EL_j = \{E_1L_j, \ldots, E_{Nj} L_j \} \cup \mathbf{\{E_{Nj+1}L_j , E_{Nj+2}L_j, \ldots, E_{Nj+Pj} L_j \}}$

Observație: Extinderea nivelurilor limbajului adaptabil poate fi reprezentat prin formarea unui dialect nou al limbajului (Modelul Horizontal).

Situația (3.d): Problema cere adăugarea unor elemente noi situate pe niveluri inferioare nivelului BAZEI limbajului adaptabil.

Soluția: Limbajul adaptabil se modifică prin adaosul elementelor nivelurilor inferioare nivelului BAZEI limbajului adaptabil: $\mathbf{EL_{0-1}, EL_{0-2}, \ldots, EL_{0-r}}$ (Modelul Top-Down).

Situația (3.e): Problema cere adăugarea unor elemente noi, suportând **cazuri mixte binare** (3.a) şi (3.b), sau (3.a) şi (3.c), sau (3.a) şi (3.d), sau (3.b) şi (3.c), sau (3.b) şi (3.d), sau (3.c) şi (3.d).

Situația (3.f): Problema cere adăugarea unor elemente noi, suportând cazuri mixte ternare.

Situația (3.g): Problema cere adăugarea unor elemente noi, suportând cazuri mixte caternare.

Soluția: Limbajul adaptabil se modifică prin adaosul elementelor noi în corespondență cu **cazurile mixte binare** (3.a) şi (3.b), sau (3.a) şi (3.c), sau (3.a) şi (3.d), sau (3.b) şi (3.c), sau (3.b) şi (3.d), sau (3.c) şi (3.d), sau în corespondență cu **situația (3.f):** cazuri mixte

ternare, sau în corespondenţă cu **situaţia (3.g):** cazuri mixte caternare.

4.4. Metodele de evaluare a sistemelor adaptabile

Sistemul adaptabil este reprezentat de limbajul şi procesorul lui adaptabil corespunzător. Procesorul adaptabil evaluează în concordanţă cu evaluarea limbajului adaptabil.

Procesorul adaptabil este compus din module translatorice. Fiecărui element din limbajul adaptabil, prezentat prin intermediul funcţiei $\varphi_{i,j,}$ i se pune în corespondenţă un modul (un lanţ de module) translatoric $\psi_{i,j}$ al procesorului adaptabil. Realizarea procesorului adaptabil se poate face dupa trei seturi de metode [Tod-05.2]: metodele **Timp-realizare** (preprocesare, interprocesare şi postprocesare), metodele **Model-realizare** (L – L (Level - Level), L – D (Level - Direct) şi L – L – D (Level - Level - Direct)) şi metodele **Tip-translatare** (compilare, interpretare, compilare-interpretare) adaptabilă.

Modificarea procesorului adaptabil depinde de situaţiile lingvistice, în care activează programatorul, susţinut de sistemul adaptabil. El este modificat în dependenţă de modificarea limbajului adaptabil. Ca şi in cazul limbajului adaptabil sunt evidenţiate situaţiile următoare.

Situaţia (4.a): Prin intermediul elementelor existente ale sistemului adaptabil pot fi realizate problemele utilizatorului. **Soluţia:** Procesorul adaptabil nu se modifică.

Situaţia (4.b): Problema nouă cere adăugarea unor elemente de nivel superior celor existente (P - numărul prezent de niveluri ale sistemului adaptabil) din sistemul adaptabil. **Soluţia:**: Se creează şi se adaogă la procesorul adaptabil un set de module translatorice noi:

$$\Psi_{1,\,p+1},\ \Psi_{2,\,p+1},\ \dots,\ \Psi_{Np+1,\,p+1}$$
$$\dots$$
$$\Psi_{1,\,p+q},\ \Psi_{2,\,p+q},\ \dots,\ \Psi_{Np+q,\,p+q}$$

Aceste module translatorice ale procesorului adaptabil permit realizarea elementelor noi adăugate la nivelurile superioare ale limbajului adaptabil. Ele permit utilizarea limbajul adaptabil modificat prin adaosul elementelor nuvelurilor superioare celor existente în limbaj: $\mathbf{EL_{p+1}, EL_{p+2}, \dots, EL_{p+q}}$.

Aceasta reprezintă o extindere *verticlală* de tip Bottom-Up (*mai sus* de BAZĂ) a sistemului adaptabil.

Situaţia (4.c): Problema cere adăugarea unor elemente noi la nivelurile deja existente ale sistemului adaptabil.

Soluţia: Procesorul adaptabil se modifică prin crearea şi adăugarea modulelor translatorice noi:

$$\Psi_{No+1,o}\ ,\ \Psi_{No+2,o},\ \dots,\ \Psi_{No+Po,o}$$
$$\dots$$
$$\Psi_{Nj+1,j}\ ,\ \Psi_{Nj+2,j}\ ,\ \dots,\ \Psi_{Nj+Pj,j}$$

Aceste module translatorice ale procesorului adaptabil permit realizarea elementelor noi adăugate la nivelurile existente respective ale limbajului adaptabil:

$$\{E_{No+1}L_o\ ,\ E_{No+2}L_o,\ \dots,\ E_{No+Po}L_o\}$$
$$\{E_{N1+1}L_1\ ,\ E_{N1+2}\ L_1,\ \dots,\ E_{N1+P1}L_1\}$$
$$\dots$$
$$\{E_{Nj+1}L_j\ ,\ E_{Nj+2}\ L_j,\ \dots,\ E_{Nj+Pj}L_j\ \}$$

Aceasta reprezintă o extindere *orizontală* a sistemului adaptabil de tip Horizontal.

Situaţia (4.d): Problema cere adăugarea unor elemente noi situate pe niveluri inferioare nivelului BAZEI sistemului adaptabil.

Soluţia: Sistemul adaptabil se modifică prin crearea şi adaugarea modulelor translatorice noi:

$$\Psi_{1,\,0\text{-}2},\ \Psi_{2,\,0\text{-}2},\ \dots,\ \Psi_{N0\text{-}2,\,0\text{-}2}$$
$$\dots$$
$$\Psi_{1,\,0\text{-}r},\ \Psi_{2,\,0\text{-}r},\ \dots,\ \Psi_{N0\text{-}r,\,0\text{-}r}$$

Aceste module translatorice ale procesorului adaptabil permit realizarea elementelor, care aparţin nivelurilor noi inferioare nivelului BAZEI, adăugate limbajului adaptabil: $\mathbf{EL_{0\text{-}1}, EL_{0\text{-}2}, \dots, EL_{0\text{-}r}}$.

Aceasta reprezintă o extindere *verticală* a sistemului adaptabil de tip Top-Down (de ex.: *mai jos* de BAZĂ).

Situaţia (4.e): Problema cere adăugarea unor elemente noi, suportând **cazuri mixte binare** (4.a) şi (4.b), sau (4.a) şi (4.c), sau (4.a) şi (4.d), sau (4.b) şi (4.c), sau (4.b) şi (4.d), sau (4.c) şi (4.d), sau echivalent cu **situaţia (**4.**f):** cazuri mixte ternare, sau echivalent cu **situaţia (**4.**g):** cazuri mixte caternare.

Soluţia: Sistemul adaptabil se modifică prin adaosul modulelor translatorice noi în corespondenţă cu **cazurile mixte binare** 4.a) şi (4.b), sau (4.a) şi (4.c), sau (4.a) şi (4.d), sau

(4.b) şi (4.c), sau (4.b) şi (4.d), sau (4.c) şi (4.d), sau echivalent cu **situaţia (**4.**f):** cazuri mixte ternare, sau sau echivalent cu **situaţia (**4.**g):** cazuri mixte caternare

Situaţia (4.**e):** Problema cere adăugarea unor module translatorice noi, suportând **cazuri mixte binare (**4.a) şi (4.b), sau (4.a) şi (4.c), sau (4.a) şi (4.d), sau (4.b) şi (4.c), sau (4.b) şi (4.d), sau (4.c) şi (4.d).

Situaţia (4.**f):** Problema cere adăugarea unor module translatorice noi, suportând cazuri mixte ternare.

Situaţia (4.**g):** Problema cere adăugarea unor module translatorice noi, suportând cazuri mixte caternare.

Soluţia: Sistemul adaptabil se modifică prin adaosul modulelor translatorice noi în corespondenţă cu **cazurile mixte binare** 4.a) şi (4.b), sau (4.a) şi (4.c), sau (4.a) şi (4.d), sau (4.b) şi (4.c), sau (4.b) şi (4.d), sau (4.c) şi (4.d), sau în corespondenţă cu **situaţia (**4.**f):** cazuri mixte ternare, sau în corespondenţă cu **situaţia (**4.**g):** cazuri mixte caternare.

5. Concluzii

Sunt cercetate sistemele (procesoarele şi limbajele) de programare tradiţionale[Tod-05.2]. Nucleul sistemelor de programare adaptabilă îl constituie adaptorii - mijloacele metalingvistice, compuse din extendori şi reductori. În baza acestor cercetări şi implementări au fost obţinute următoarele rezultate:

Este realizată experimental una din multiplele scheme realizatorice ale cubului adaptabilităţii, prezentată în formă de sistem adaptabil SAGRED3D [Mic-02], care se utilizează în procesul de învăţământ cu studenţii facultăţii de matematică şi informatică a U.S.M. (Chişinău) şi cu studenţii facultăţii de informatică a Universităţii "Al.I.Cuza" (Iaşi).

(1) Sunt cercetate integral facilităţile extensibilo-adaptabile ale mijloacelor adaptabile din punct de vedere lingvistic şi de sistem;

(2) Sunt cercetate şi analizate funcţional schemele de preprocesor, interprocesor şi postprocesor ca reprezentanţi principiali pe una din dimensiunile cubului adaptabilităţii; o altă dimensiune din acest cub de procesoare adaptabile cercetate şi analizate o constituie procesoarele: Nivel-Nivel, Nivel-Direct şi Nivel-Nivel-Direct;

(3) Procesul de extindere este reprezentat prin intermediul sistemelor de Declarare, Fixare şi Utilizare. Procesul de reducere este prezentat de sistemul de Reducere. În teză aceste subsisteme sunt analizate din punct de vedere a creării procesoarelor adaptabile de diferit tip. Procesele de extindere şi reducere împreună formează procesul de adaptare;

(4) Au fost formulate şi demonstrate [Mic-03.1] teoremele de obţinere automatizată a procesoarelor adaptabile de nivelul (5.1) unu (de tip model-realizare: Nivel-Nivel, Nivel-Direct şi Nivel-Nivel-Direct, de tip timp-realizare: preprocesor, interprocesor şi postprocesor şi de tip de translatare: compilator, interpretor şi compilator-interpretor), (5.2) doi (de tip Timp-Model-Realizare) şi (5.3) trei (de tip Timp-Model-Realizare cu compilare) de complexitate translatorică. Sa obţinut fundamentarea teoretică a algoritmilor propuşi de soluţionare a problemelor cercetate.

Un set important de probleme, aflate la etapa de soluţionare în vederea evaluării programării adaptabile îl constituie:

(5.1) Timpul (metodele timp-realizare) de realizare a extensiilor: metodele de preprocesare, interprocesare, postprocesare şi/sau mixt;

(5.2) Tipul de translatare adaptabilă: metodele de compilare, interpretare şi/sau mixt;

(5.3) Modelul de adaptare multilingvistică: metodele Top–Down, Bottom–Up, Horizontal şi/sau mixt;

(5.4) Metoda model-realizare a extensiilor: Nivel-Nivel, Nivel-Direct, Nivel-Nivel-Direct şi/sau mixt.

În baza lucrării [Tod-05.2] în prezenta lucrare de continuare a cercetărilor în domeniul adaptabilităţii sunt analizate şi dezvoltate problemele extensibilităţii limbajelor prin definiţii de extensii de limbaj.

Sunt prezentate extensii de date, operaţii, instrucţiuni şi dirijări.

Sunt enumerate un set de proiecte de sisteme extensibile şi adaptabile.

Sunt analizate metodele de evaluare a limbajelor şi sistemelor adaptabile: metodele lin-

gvistice BOTTOM-UP, TOP-DOWN şi HORIZONTAL, de asemenea şi metodele evaluate de creare a limbajelor adaptabile şi a sistemelor adaptabile.
Sunt prezentate rezultatele de generalizare a noţiunii adaptabilităţii începând cu Baza adaptabilă şi continuând cu nivelurile limbajelor adaptabile.

Bibliografie

[Tod-06] D. Todoroi, T. Jucan, D. Micusha. Adaptable Systems. // 2006, To be published. (Eng.)
[Tod-05.1] Dumitru TODOROI, Diana MICUSA, Zinaida TODOROI. Adaptable Languages and Systems Properties. // Proc. of the 7th Int. Conf. on informatics in Economy "Information & Knowledge Age", Bucharest, May 19-21, 2005, pp. 349-355.
[Tod-05.2] Dumitru TODOROI. Sisteme adaptabile. Facilitati lingvistice si de system. // Analele ASEM, Editia a III-a, 2005, pp. 186-196.
[Tod-05.3] Dumitru TODOROI, Diana MICISA, Zinaida TODOROI. Adaptable Systems Linguistic and Processing Properties. // Informatica Economica, Vol. IX, Nr.3(35)/2005, INFOREC, Bucutesti, pp. 109-117.
[Tod-05.4] Dumitru TODOROI. The Theoretical Bases of **the first level** of Adaptable Processors creation. //Proc. of the 30th Annual Congress of the ARA, Central Publication House, Chişinău, 2005, pp. 178-191.
[ZTod-05] Zinaida Todoroi, Manfred Kudlek, Diana Micusa, Dumitru Todoroi. Crearea automatizată a procesoarelor adaptabile de **nivelul doi** bazate pe nivelul unu cu utilizarea T-formalismului de interacţiune translatorică. // Informatica Economica, Vol. IX, Nr.1(33)/2005, INFOREC, Bucutesti, pp. 47-55.
[Tod-04] D Todoroi, D. Micusa. Sisteme adaptabile. Definitii. Moltilingvibilitatea. // Simp. International « Integrarea europeana si competivitatea economica », 23-24 septembrie 2004, Vol. II, Chisinau 2004, pp. 217-220.
[Mic-04] D. Micusa, D Todoroi. Sisteme adaptabile. Proprietati lingvistice si de sistem. // Simp. International « Integrarea europeana şi competivitatea economică », 23-24 septembrie 2004, Vol. II, Chişinău 2004, pp. 221-223.
[Mic-03] Diana Micusa, Manfred Kudlek, Dumitru Todoroi, Zinaida Todoroi. Crearea automatizată a procesoarelor adaptabile de nivelul trei în baza formalismului E-T-M de interactiune translatorica a procesoarelor adaptabile. // Informatica economica, Vol. VII, Nr. 2(26)2003, INFOREC, Bucuresti, pp.103-109.
[Tod-98] D. Todoroi, S. Nazem, T. Jucan, D. Micusha. Transition to a Full Information Society: Stage Development. // Working Paper No. 98-2, UNO, Omaha, USA, March 1998, 38 pages.
[Tod-92] D. Todoroi. Computer Science. The Adaptable Programming. The Basic Conceptions. // ASEM Press, Chisinau, 1992. - 76 pages. (Eng.)
[Cuz-89] Кузнецов С., Тодорой Д.. О школе семинаре «Расширяемые средства программирования. Методы оценки трансляторов». // Программирование. 1989. №6. pp.91-92.
[Tod-87] Тодорой Д., Челбаков Б. Целевая программа «Расширяемые средства программирования». // Программирование. 1987, №3, pp. 91-92.
[Tod-83] D. Todoroi. The Extensible Computer Graphics Tools. // Radio and Telecommunication Press, Moscow, 1983. - 208 pages. (Rus.)
[Tod-80] D.Todoroi. The Extensible Computer Graphics Systems. // Science Press, Chisinau, 1980, 199 pages. (Rus.)
[Sol-74] N. SOLNTSEFF, A. YEZERSKI. A Survey of Extensible Programming Language. // ARAP, 1974, 4, pp.267-307.
[Pro-71] Proceedings of Extensible Languages Symposium. // SIGPLAN Notices. December, 1971, 6, Nr.12.
[Pro-69] Proceedings of the Extensible Languages Symposium (Boston, May, 1969) Eds C. Christiansen. // SIGPLAN Notices, 1969, v.4, Nr.8.
[Tod-68.1] D. Todoroi. The Recursive Programs' ALGOL - 60 implementation. Parts 1, 2. // Programming Automation, Academy Press, Kiev, 1968, pp.18 - 55. (Rus.)
[Tod-68.2] D. Todoroi. The Small Recursive Translator. // Application Software, Science Press, Chisinau, 1968, pp. 2-71. (Rus.)
[Tod-67] D. Todoroi. "Sistemul de implementare a programelor recursive: SIRP" . // SoftPress GIPROTIS, Moscova, 1967, 38 pages (Rus)