

Processing and Linguistics Properties of Adaptable Systems (Part 2)

Dumitru TODOROI, Diana MICUȘA, Zinaida TODOROI
Academia de Studii Economice din Moldova

Continuation and development of the research in Adaptable Programming Initialization [Tod-05.1,2,3] is presented. As continuation of [Tod-05.2,3] in this paper metalinguistic tools used in the process of introduction of new constructions (data, operations, instructions and controls) are developed. The generalization schemes of evaluation of adaptable languages and systems are discussed. 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]. Presented results will be included in the book [Tod-06].

Introducere

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

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

Primele două forumuri internaționale, consa-

crate în special limbajelor și sistemelor extensibile [Pro-69,71] au evidențiat proprietățile și facilitățile limbajelor și sistemelor extensibile și au accentuat problemele, care apar în această direcție de cercetări și implementări a sistemelor de interacțiune om-mașină. Prima generalizare a rezultatelor, obținute în domeniul extensibilității, a fost efectuată de Solntseff și Yezerski [Sol-74] din punct de vedere al divizării mijloacelor extensibile în dependență de fazele de translatare. În țările FUS și 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 și adaptabilității din punct de vedere lingvistic, sistemic, al realizării și 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 - și acțiune-orientată. Realizările RAB, IADRO, SAGRED3D, C++, Ada și Java sunt confirmări practice ale teoriei și practicii programării adaptabile. În această lucrarea – continuare a lucrării [Tod-05.2,3] vor fi analizate și dezvoltate problemele extensibilității limbajelor prin definiții de extensii de limbaj. Sunt prezentate extensii de date, operații, instrucțiuni și dirijări. Sunt enumerate un set de proiecte de sisteme extensibile și adaptabile. Sunt prezentate rezultatele de generalizare a noțiunii adaptabilității începând cu Baza adaptabilă și continuând cu nivelurile limbajelor adaptabile. Sunt

analizate metodele de evaluare a limbajelor și sistemelor adaptabile: metodele lingvistice BOTTOM-UP, TOP-DOWN și HORIZONTAL, de asemenea și metodele evaluate de creare a limbajelor adaptabile și a sistemelor adaptabile.

1. Adaptabilitate și extensibilitate.

Cum este evidențiat în [Tod-05.2], mijloacele metalingvistice, destinate introducerii de noi construcții (date, operații, instrucțiuni, dirijări) în limbaj se numesc **extensii de limbaj**.

Aceste extensii sunt elemente de limbaj și sunt părți componente ale limbajului – Baza adaptabilă. Extensia de limbaj este realizată prin intermediul **definitivului** și **apelului extensiei**. **Definitivul** de extensie permite extinderea limbajului cu noi elemente derivate ale limbajului. El realizează definirea extensiei. **Definitivul** de extensie la fel ca și **definitivul** de reducere (individualizare) includ: definirea pragmaticei 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 **definitiv de extindere (Adaptor-extendor)** și **definitiv de reducere (Adaptor-reductor)**.

Schema generală a **adaptorului-extendor** este următoarea:

BL pragmaticele elementului-extensie
 SY sintaxa elementului-extensie
 SE semantica elementului-extensie
 CO contextul de utilizare a elementului
 EX exemple de apel al elementului-extensie
 EL

Din definiție reiese, că părțile componente ale **definitivului elementului-extensie** [Tod-05.2], sunt pragmaticele, sintaxa, semantica, mediul de utilizare și exemplarele de apel a noului element definit.

Pragmatica identifică locul, necesitatea extensiei în limbaj, explică necesitatea includerii în limbaj a elementului nou. **Sintaxa** definește forma de apel a extensiei – elementului nou. **Semantica** reprezintă descrierea conținutului elementului nou prin intermediul conținutului altor elemente deja introduse în limbaj. **Contextul** apreciază locul de utilizare a acestor elemente derivate, ierarhia lor în cadrul elementelor limbajului. **Exemplele**

conțin exemple de apel și de utilizare a elementului nou introdus.

2. Extensii de limbaj

Utilizarea **adaptorului-extendor** este efectuată prin intermediul utilizării modificărilor lui la prezentarea exemplelor de extensii de date, operații, instrucțiuni și dirijări. Prezentarea fiecărei din extensii este precedată de prezentarea mediului operațional, în care această extensie va avea loc. Extensia este prezentată prin intermediul schemei generale a **adaptorului-extendor** evaluată pentru cazurile de prezentare a extensiilor de date, operații, instrucțiuni și dirijări.

2.1. Extensii de date

Dorim să declarăm o dată nouă cu numele de `<vector>`. Această dată va fi exprimată printr-un element de nivel inferior cu numele de `<ARRAY>`.

Mediul operațional de definire a elementului-extensie-dată `<vector>`. Sunt cunoscute: (1) noțiunile neterminale: `<type>`, `<identif>`, `<bj>` (bariera de jos), `<bs>` (bariera de sus), (2) simbolurile metalingvistice: `DBL`, `DSY`, `DES`, `DEX`, `DEL` și `AS` și (3) elementele terminale B, C, 10, 1, 200, ARRAY, [,], -, :, < și >.

În acest mediu noul element-extensie de date `<vector>` va fi definit prin intermediul **adaptorului-extendor de date** în forma următoare:

```
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” și este definit prin intermediul elementelor de nivel mai jos “i”: ARRAY, `<identif>`, `<bj>`, `<bs>`, și altele.


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 operații




De definit o nouă operație de sumare a vectorilor cu numele de `<sum vector>`. **Mediul operațional de definire** a elementului-extensie-operație `<sum vector>`. Sunt cunoscute: (1) operația “+” (sumare), (2) noțiunile de identificatori: `<identif1>` și `<identif2>`, (3) instrucțiunile de atribuire, înscriere a rezulta-

tului și cele de dirijare a ciclului. Sunt cunoscute sistemului adaptabil de asemenea: (4) funcția **Length**, (5) metaterminalele: `ˆOBL`, `ˆOCO`, `ˆAS`, `ˆOSY`, `ˆOSE`, `ˆOEX` și `ˆOEL` și (6) terminalele: A, M, 1, k, x, y, P, Q, VECTOR, CYCLE, FROM, TO, BYSTEP, DO, ENDCYCLE, RESULT, <, >, [,], (,), +, :, ;, := și =.

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

```
ˆOBL ˆ<sum vector>
ˆOCO ˆAS ˆ<sumare>
ˆOSY ˆ<identif1>  ˆ<identif2>
```

```
ˆOSE A := Length (<identif1>);
      k : VECTOR [1:A];
      CYCLE M FROM 1 TO A BYSTEP 1 DO
          k[M]:= <identif1> [M]+ <identif2> [M];
      ENDCYCLE;
      RESULT VECTOR;
```

```
ˆOEX 
      x  y
      P  Q
```

```
ˆOEL
```

Pentru această definiție se presupune, că numărul elementelor vectorilor cu numele de `<identif1>` și de `<identif2>` sunt egale.

2.3. Extensii de instrucțiuni

De definit o nouă instrucțiune cu numele de `<Lin-Frântă>`. **Mediul operațional de definire** a elementului-extensie-instrucțiune `<Lin-Frântă>`. Sunt cunoscute: (1) datele de tip ARRAY, (2) instrucțiunile `<ciclu>`, `<punct>`, `<segment>` și `<declarare>`, (3) funcția **Length** și tipul **INT**, (4) metadelimitatorii: `ˆSBL`, `ˆSCO`, `ˆAS`, `ˆSSY`, `ˆSSE`, `ˆSEX`, `ˆSEL` și `ˆKIND` și (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.

```
ˆCBL ˆ<cycle>
ˆCCO ˆAS ˆ<statement>
ˆCSY ˆKIND:1: WHILE <exp2> DO <statement>;
      KIND:2: FOR <ep>:= <ae1> TO <ae2> BY.STEP <ae3> DO <statement>;
ˆCSE ˆKIND: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: ;
ˆCEX 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;
ˆCEL
```

A fost definită o nouă extensie-dirijare cu numele de `<Cycle>`. Ea este definită în două forme: WHILE-DO și FOR-DO. Semantica acestor doua elemente-dirijări este definită prin intermediul dirijărilor de un nivel mai

În acest mediu noul elementul-extensie de instrucțiune `<Lin-Frântă>` va fi definit prin intermediul **adaptorului-extendor de instrucțiuni** în forma următoare:

```
ˆSBL ˆ<Lin-Frântă>
ˆSCO ˆAS ˆ<punct>, <ciclu>, <segment>
ˆSSY ˆKIND: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: INT);
ˆSSE ˆKIND: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]);
      ENDCYCLE
ˆSEX BROKEN.LINE (T,S);
      BROKEN.LINE (kx: ARRAY [1:k], ky:ARRAY [1:k])
      BROKEN.LINE (kTX, kTY: ARRAY; P:INTEGER)
ˆSEL
```

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

2.4. Extensii de dirijări

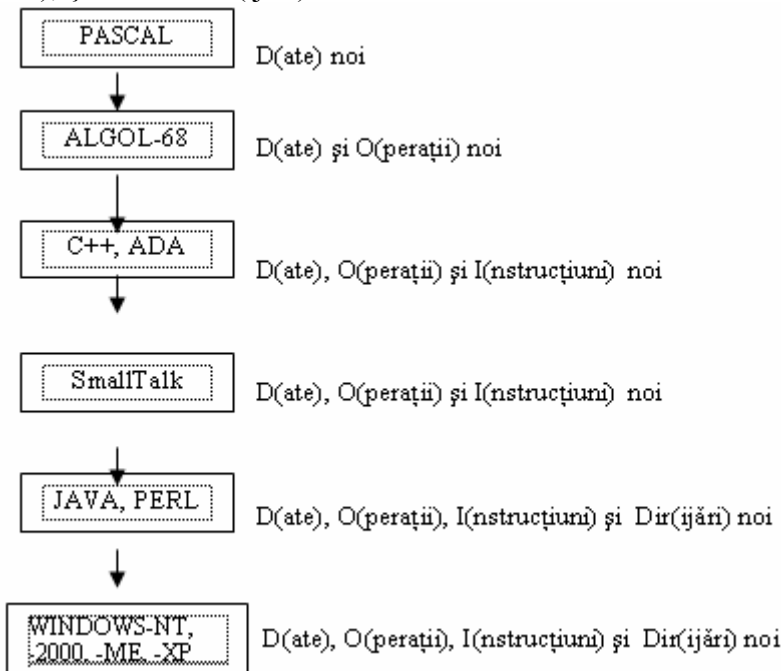
De definit o nouă dirijare cu numele de `<Cycle>`. **Mediul operațional de definire** a elementului-extensie-dirijare `<Cycle>`. Sunt constituie următoarele elemente: (1) expresia `<exp2>` și instrucțiunea `<statement>`, (2) expresiile `<ep>`, `<ae1>`, `<ae2>`, `<ae3>`, (3) instrucțiunile `<compusă>` și `<atribuire>`, (4) dirijările `<conditional>` și `<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 și END și (6) metaterminalele: `ˆCBL`, `ˆCCO`, `ˆAS`, `ˆCSY`, `ˆCSE`, `ˆCEX`, `ˆCEL` și `ˆKIND`. În acest mediu noul elementul-extensie de instrucțiune `<Cycle>` va fi definit prin intermediul **adaptorului-extendor de dirijări** în forma următoare:

jos: IF-THEN și GOTO și instrucțiunea compusă BEGIN-END.

3. Evaluarea limbajelor si sistemelor extensibile și adaptabile

Extendorii și în continuare adaptorii [Tod-04,

Mic-04, Tod-92] au participat la crearea și realizarea unei serii de proiecte și sisteme adaptabile și extensibile reale și/sau experimentale. Aceste limbaje și sisteme extensibile-adaptabile se caracterizează prin extinderea seturilor lor de D(ate), și/sau O(perații), și/sau I(nstrucțiuni), și/sau de Dir(ijări). Cele



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.

Dirijările externe între diferite programe și între mașini le constituie metodele de multiprogramare și multiprocesare, executarea virtuală a programelor, sistemele de divizare a timpului etc.

Exemple de dirijări externe (multiprocesare): mașini vectoriale, mașini 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 mașini de generația I-a și a II-a. El a fost pe deplin modular și destul de efectiv și eficient din punct de vedere a timpului de execuție a programelor ALGOL și a

mai reprezentative limbaje și sisteme extensibile-adaptabile în evoluția limbajelor și sistemelor de programare sunt: PASCAL, ALGOL-68, C++, ADA, SmallTalk, JAVA, PERL, WINDOWS-NT, WINDOWS-2000, WINDOWS-ME, WINDOWS-XP.

memoriei ocupate în comun de translator și programul-obiect obținut de translator. Motorul acestui translator l-a constituit “Sistemul de implementare a programelor recursive: SIRP” [Tod-68.2], care a fost partea componenta a translatorului ALGOL-60.

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

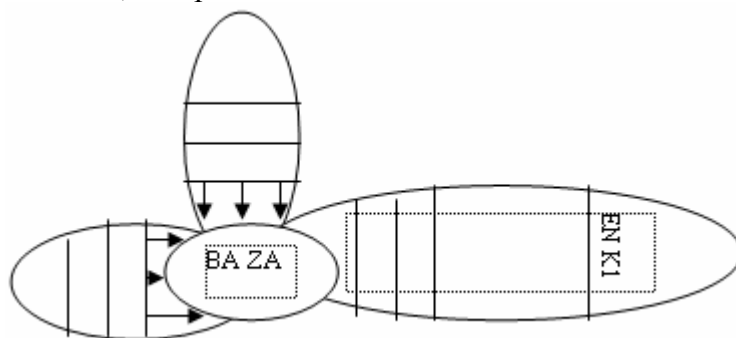
- Sistema universală extensibilă “MACRO SYSTEMA-77” (anul de dare în exploatare: 1977)
- Baza algoritmică extensibilă “RAB” (anul de dare în exploatare: 1980)
- Sistema extensibilă de grafică pe calculator “YADRO” (anul de dare în exploatare: 1985)
- Sistema adaptabilă universală “COMSYBASIC” (anul de dare în exploatare: 1988)
- Proiectul “PRARM” (anul de dare în evaluare: 1990)
- Sistema extensibilă de grafică pe calculator “CRED3D” (anul de dare în exploatare: 1992)
- Sistema adaptabilă universală “SAGRE D3D”(anul de dare în exploatare: 1997)

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

șaptezeci [Ear-70, Sol-74] și până în prezent [Tod-92, Tod-04] cu concursul multor metode variate de interacțiuni translatorice [Mic-04,03]. O bibliografie imensă de idei și realizări de limbaje și sisteme orientate obiect, orientate agent, extensibile și adaptabile, care sunt bazate pe ideea extensibilității și adaptabilității, poate fi selectată doar enumerând comunicările, publicate în cadrul ultimelor 20 de ani de conferințe anuale internaționale OOPSLA.

4. Generalizări

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



Baza este compusă din extendori, reductori și elementele de bază, incluse axiomatice în limbajul adaptabil și sistemul respectiv. Elementele de nivel sunt definite prin intermediul extendorilor și reductorilor. Prin intermediul **extendorului** (definiție și apel) se obține extinderea limbajului și a sistemului adaptabil. Această operație poate fi efectuată prin intermediul extinderii de un anumit tip: date, instrucțiune, operație sau dirijare. Prin extinderi succesive se ajunge de la bază la nivelul limbajului adaptabil necesar $EN K_i$, scopul final fiind obținerea limbajului natural de comunicare formalizat.

Prin intermediul **reductorului** se obține un sistem de extinderi individuale. În acest caz utilizatorul evidențiază elementele necesare pentru sistemul adaptabil de programare individual, tipul procesoarelor, care vor activa în acest sistem individual și formele de trecere dintr-un nivel în altul. Rezultatul acestei

reduceri îl constituie sistemul adaptabil individual. Acest sistem va fi compus din baza extensibilă și elemente de nivel deja definite în acest sistem adaptabil. Se pot efectua extinderi și reduceri în diferite direcții. Sistemul adaptabil este un sistem universal și specializat. El dispune de multe alte proprietăți [Tod-05.2].

4.1. Baza și 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 $E_i L_j$. Elementul $E_i L_j$ este definit ca o funcție:

$$E_i L_j = \varphi_{ij}(T, EL_0, EL_1, \dots, EL_{j-1}, EL_j', EL_j''),$$

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 – numărul de elemente ale nivelului “j”,

(5) $EL_k = (E_1L_k, E_2L_k, \dots, E_{N_k}L_k)$ – setul de elemente ale nivelului “k” și

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

În procesul de definire a elementelor este necesar de evidențiat **situațiile** suplimentare:

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

(1.b) Setul de elemente $EL_j'' = (E_1L_j'', \dots, E_{N_j}''L_j'')$ reprezintă mulțimea elementelor nivelurilor superioare elementului E_iL_j , prin intermediul cărora sunt definite elementele acestui nivel “j” și printr-o posibilă recursie indirectă.

(1.c) Funcția φ_{ij} este funcția de realizare a elementului “i” de pe nivelul “j” cu ajutorul a cel puțin a unui element de pe nivelul “j-1”, unde $i = 1, 2, \dots, N_j$ și $j = 1, 2, \dots, P$;

(1.d) BAZA adaptabilă: elementele E_iL_0 din EL_0 , unde $i = 1, 2, \dots, N_0$ și 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:

$\langle \text{expresia aritmetică primară} \rangle ::= \langle \text{constantă} \rangle \mid \langle \text{variabilă} \rangle \mid \langle \text{apel de funcție} \rangle \mid (\langle \text{expresia aritmetică} \rangle) : \text{situația (1.b): recursie indirectă}$

$\langle \text{factor} \rangle ::= \langle \text{expresia aritmetică primară} \rangle \mid \langle \text{factor} \rangle \uparrow \langle \text{expresia aritmetică primară} \rangle : \text{situația (1.a): recursie directă}$

$\langle \text{term} \rangle ::= \langle \text{factor} \rangle \mid \langle \text{term} \rangle * \langle \text{factor} \rangle : \text{situația (1.a): recursie directă}$

$\langle \text{expresia aritmetică simplă} \rangle ::= \langle \text{term} \rangle \mid \langle \text{expresia aritmetică simplă} \rangle + \langle \text{term} \rangle : \text{situația (1.a): recursie directă}$

$\langle \text{expresia aritmetică} \rangle ::= \langle \text{expresia aritmetică simplă} \rangle \mid \langle \text{expresia aritmetică condiționa-$



Nivel apropiat de baza grafică a limbajului adaptabil



Nivel intermediar



Nivelul cel mai jos de comunicare grafică

lă>

4.2. Metodele adaptabile de evaluare procesorală

Există trei tipuri de creștere a limbajelor și sistemelor adaptabile: creștere în sus (spre utilizator) – metoda adaptabilă procesorală BOTTOM-UP, creștere în jos (spre Bază) – metoda adaptabilă procesorală TOP-DOWN, creștere orizontală – metoda adaptabilă procesorală HORIZONTAL.

4.2.1. Metoda adaptabilă procesorală BOTTOM-UP. Creșterea 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 creșterii limbajului până la obținerea sistemului adaptabil, orientat al utilizatorului. Acest proces se numește **creșterea BOTTOM-UP** a limbajului adaptabil.

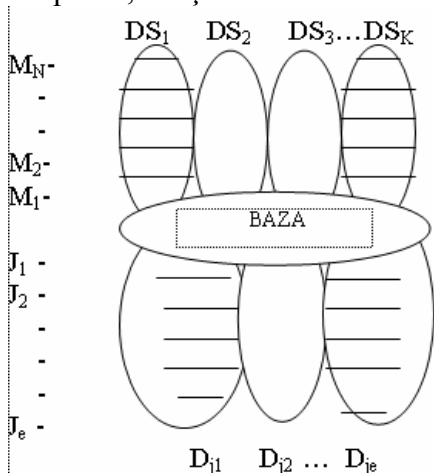
4.2.2. Metoda adaptabilă procesorală TOP-DOWN. Creșterea în jos a limbajului adaptabil de asemenea este efectuează cu ajutorul extendorilor. Efectul acestei evaluări constă în aceea, că elementele de bază sunt extinse spre diferite dispozitive ale calculatorului. Acest proces se numește **creșterea TOP-DOWN** a limbajului adaptabil.

De exemplu, fie dat un set de dispozitive grafice cu diferite niveluri ale limbajelor de programare din grafica pe calculator: analitică și reprezentativă. Unele grafplotere, ca dispozitive anexate la calculator, pot doar să apropie creionul de hârtie, să-l ridice și să-l miște 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. În ordinea descreșterii “intelectualității” dispozitivelor, ele au următoarele posibilități:

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 “intelect” foarte redus (nivelul cel mai jos de comunicare grafică).

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

În așa fel se obțin dialectele de sus ale limbajului adaptabil, compuse din elemente derivate după metoda Bottom-Up de creștere a limbajului: DS₁, DS₂, ..., DS_K și 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 și prin intermediul ei se îmbogățesc atât nivelurile dialectelor de sus și de jos ale limbajului adaptabil cât și baza. Schema limbajului adaptabil, în așa 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 (P - numărul prezent de niveluri ale limbajului adaptabil) în limbajul adaptabil. Soluția: Limbajul adaptabil se modifică prin adaosul elementelor nivelurilor superioare celor existente în limbaj: EL_{p+1}, EL_{p+2}, ..., 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:

$$EL_0 = \{E_1L_0, \dots, E_{N_0}L_0\} \cup \{E_{N_0+1}L_0, E_{N_0+2}L_0, \dots, E_{N_0+P_0}L_0\}$$

$$EL_1 = \{E_1L_1, \dots, E_{N_1}L_1\} \cup \{E_{N_1+1}L_1, E_{N_1+2}L_1, \dots, E_{N_1+P_1}L_1, \dots\}$$

$$EL_j = \{E_1L_j, \dots, E_{N_j}L_j\} \cup \{E_{N_j+1}L_j, E_{N_j+2}L_j, \dots, E_{N_j+P_j}L_j\}$$

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: EL₀₋₁, EL₀₋₂, ..., EL_{0-r} (Modelul Top-Down).

Situația (3.e): Problema cere adăugarea unor elemente noi, suportând cazuri mixte binare (3.a) și (3.b), sau (3.a) și (3.c), sau (3.a) și (3.d), sau (3.b) și (3.c), sau (3.b) și (3.d), sau (3.c) și (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) și (3.b), sau (3.a) și (3.c), sau (3.a) și (3.d), sau (3.b) și (3.c), sau (3.b) și (3.d), sau (3.c) și (3.d), sau în corespondență cu **situația (3.f)**: cazuri mixte ternare, sau în corespondență cu situația (3.g): cazuri mixte caternare.

4.4. Metodele de evaluare a sistemelor adaptabile

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

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

Modificarea procesorului adaptabil depinde de situațiile lingvistice, în care activează programatorul, susținut de sistemul adaptabil. El este modificat în dependență de modificarea limbajului adaptabil. Ca și în cazul limbajului adaptabil sunt evidențiate situațiile următoare.

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

Situația (4.b): Problema nouă cere adăugarea unor elemente de nivel superior celor existente (P - numărul prezent de niveluri ale sistemului adaptabil) în sistemul adaptabil. **Soluția**: Se creează și se adaugă la procesorul adaptabil un set de module translatorice noi:

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

...

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

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

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

Situația (4.c): Problema cere adăugarea unor elemente noi la nivelurile deja existente ale sistemului adaptabil. **Soluția**: Procesorul adaptabil se modifică prin crearea și adăugarea modulelor translatorice noi:

$$\Psi_{N_0+1,0}, \Psi_{N_0+2,0}, \dots, \Psi_{N_0+P_0,0}$$

...

$$\Psi_{N_j+1,j}, \Psi_{N_j+2,j}, \dots, \Psi_{N_j+P_j,j}$$

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

$$\{E_{N_0+1}L_0, E_{N_0+2}L_0, \dots, E_{N_0+P_0}L_0\}$$

$$\{E_{N_1+1}L_1, E_{N_1+2}L_1, \dots, E_{N_1+P_1}L_1\}$$

$$\{E_{N_j+1}L_j, E_{N_j+2}L_j, \dots, E_{N_j+P_j}L_j\}$$

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

Situația (4.d): Problema cere adăugarea unor elemente noi situate pe niveluri inferioare nivelului BAZEI sistemului adaptabil. **Soluția**: Sistemul adaptabil se modifică prin crearea și adăugarea modulelor translatorice noi:

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

...

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

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

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

Situația (4.e): Problema cere adăugarea unor elemente noi, suportând **cazuri mixte binare** (4.a) și (4.b), sau (4.a) și (4.c), sau (4.a) și (4.d), sau (4.b) și (4.c), sau (4.b) și (4.d), sau (4.c) și (4.d), sau echivalent cu **situația (4.f)**: cazuri mixte ternare, sau echivalent cu **situația (4.g)**: cazuri mixte caternare. **Soluția**: Sistemul adap-

tabil se modifică prin adaosul modulelor translatorice noi în corespondență cu **cazurile mixte binare** 4.a) și (4.b), sau (4.a) și (4.c), sau (4.a) și (4.d), sau (4.b) și (4.c), sau (4.b) și (4.d), sau (4.c) și (4.d), sau echivalent cu **situația (4.f)**: cazuri mixte ternare, sau echivalent cu **situația (4.g)**: cazuri mixte caternare

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

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

Soluția: Sistemul adaptabil se modifică prin adaosul modulelor translatorice noi în corespondență cu **cazurile mixte binare** 4.a) și (4.b), sau (4.a) și (4.c), sau (4.a) și (4.d), sau (4.b) și (4.c), sau (4.b) și (4.d), sau (4.c) și (4.d), sau în corespondență cu **situația (4.f)**: cazuri mixte ternare, sau în corespondență cu **situația (4.g)**: cazuri mixte caternare.

5. Concluzii

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

Este realizată experimental una din multiplele scheme realizatoare ale cubului adaptabilității, prezentată în formă de sistem adaptabil SAGRED3D [Mic-02], care se utilizează în procesul de învățământ cu studenții facultății de matematică și informatică a U.S.M. (Chișinău) și cu studenții facultății de informatică a Universității "Al.I.Cuza" (Iași).

(1) Sunt cercetate integral facilitățile extensibil-adaptabile ale mijloacelor adaptabile din punct de vedere lingvistic și de sistem;

(2) Sunt cercetate și analizate funcțional schemele de preprocesor, interprocesor și postprocesor ca reprezentanți principali pe una din dimensiunile cubului adaptabilității; o altă dimensiune din acest cub de procesoare adaptabile cercetate și analizate o constituie procesoarele: Nivel-Nivel, Nivel-Direct și Nivel-Nivel-Direct;

(3) Procesul de extindere este reprezentat prin intermediul sistemelor de Declarație, Fixare și 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 și reducere împreună formează procesul de adaptare;

(4) Au fost formulate și demonstrate [Mic-03.1] teoremele de obținere automatizată a procesoarelor adaptabile de nivelul (5.1) unu (de tip model-realizare: Nivel-Nivel, Nivel-Direct și Nivel-Nivel-Direct, de tip timp-realizare: preprocesor, interprocesor și postprocesor și de tip de traducere: compilator, interpretor și compilator-interpretor), (5.2) doi (de tip Timp-Model-Realizare) și (5.3) trei (de tip Timp-Model-Realizare cu compilare) de complexitate translatorică. S-a obținut fundamentarea teoretică a algoritmilor propuși de soluționare a problemelor cercetate.

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

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

(5.2) Tipul de traducere adaptabilă: metodele de compilare, interpretare și/sau mixt;

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

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

În baza [Tod-05.2,3] prin prezenta lucrare de continuare a cercetărilor în domeniul adaptabilității sunt analizate și dezvoltate problemele extensibilității limbajelor prin definiții de extensii de limbaj.

Sunt prezentate extensii de date, operații, instrucțiuni și dirijări.

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

Sunt analizate metodele de evaluare a limbajelor și sistemelor adaptabile: metodele lingvistice BOTTOM-UP, TOP-DOWN și HORIZONTAL, de asemenea și metodele evaluate de creare a limbajelor adaptabile și a sistemelor adaptabile.

Sunt prezentate rezultatele de generalizare a noțiunii adaptabilității începând cu Baza adaptabilă și continuând cu nivelurile limbajelor adaptabile.

Bibliografie

[Tod-06] D. Todoroi, T. Jucan, D. Micusha. Adaptable Systems. // 2006, To be published.

[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, p. 186-196.

[Tod-05.3] Dumitru TODOROI, Diana MICUSA, Zinaida TODOROI. Adaptable Systems Linguistic and Processing Properties. // Informatica Economica, Vol. IX, Nr.3(35)/2005, INFOREC, Bucutesti, p. 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, Chișinău, 2005, p. 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 interacțiune translatorică. // Informatica Economica, Vol. IX, Nr.1(33)/2005, INFOREC, Bucutesti, p. 47-55.

[Tod-04] D Todoroi, D. Micusa. Sisteme adaptabile. Definitii. Multilingvibilitatea. // Simp. International « Integrarea europeana si competitivitatea economica », 23-24 septembrie 2004, Vol. II, Chisinau 2004, p. 217-220.

[Mic-04] D. Micusa, D Todoroi. Sisteme adaptabile. Proprietati lingvistice si de sistem. // Simp. International « Integrarea europeana si competitivitatea economica », 23-24 septembrie 2004, Vol. II, Chisinau 2004, p. 221-223.

[Mic-03] Diana Micusa, Manfred Kudlek, Dumitru Todoroi, Zinaida Todoroi. Crearea automatizată a procesoarelor adaptabile de **nivelul trei** in baza formalismului E-T-M de interactiune translatorica a procesoarelor adaptabile. // Informatica economica, Vol. VII, Nr. 2(26)2003, INFOREC, Bucuresti,

p.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 p.

[Tod-92] D. Todoroi. Computer Science. The Adaptable Programming. The Basic Conceptions. // ASEM Press, Chisinau, 1992. - 76 p. (Eng.)

[Cuz-89] Кузнецов С., Тодорой Д.. О школе семинаре «Расширяемые средства программирования. Методы оценки трансляторов». // Программирование. 1989. №6. стр.91-92.

[Tod-87] Тодорой Д., Челбаков Б. Целевая программа «Расширяемые средства программирования». // Программирование. 1987, №3, стр. 91-92.

[Tod-83] D. Todoroi. The Extensible Computer Graphics Tools. // Radio and Telecommunication Press, Moscow, 1983. - 208 p. (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, p.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.

D. Todoroi. The Recursive Programs' ALGOL - 60 Implementation. Parts 1, 2. // Programming Automation, Academy Press, Kiev, 1968, pp.18 - 55. (Rus.)

D. Todoroi. The Small Recursive Translator. // Application Software, Science Press, Chisinau, 1968, pp. 2-71. (Rus.)