METODOLOGII DE MODELARE ŞI PARADIGME DE PROGRAMARE

Autor: Ivan ZAREA
Conducător ştiinţific: mag. Radu MELNIC

Universitatea Tehnică a Moldovei

Abstract: *În acest articol se examinează relaţia dintre metodologiile de modelare şi paradigma, în cadrul căreia va fi implementat proiectul modelat. Se prezintă definiţia de bază utilizată în metodologii de modelare şi se discută aplicabilitatea universală a ei.*

Cuvinte cheie: *modelare, paradigme, Booch, UML, programare obiect-orientată, programare funcţională.*

1. Introducere

Dezvoltarea pilotată de modelare (model-driven development) e un mod de abordare a procesului de dezvoltare a produselor program care confruntă inabilitatea limbajelor de nivel înalt de a evita complexitatea platformelor şi de a exprima concepte a domeniului într-un mod efectiv [1]. Modelarea, fiind un instrument de o utilitate mare în timpul proiectării unui program, a ajuns la o etapă la care nu include aspecte de generare a codului sau de transformare a modelelor dintre diverse implementări în specificarea sa.

2. Nivele de abstracţie

Metamodelele, entităţile limbajelor de modelare care operează cu alte entităţi ale aceluiaşi limbaj (pachetele, comentarii) pot fi grupate în 4 categorii, din punct de vedere al teoriei mulţimilor [2]:

- O mulţime de clase (M_C);
- O mulţime de moşteniri M_M, reprezentată de tupluri de elemente din M_C; $M_M \subset (M_C, M_C)$;
- O mulţime de asocieri M_A, reprezentată, lafel, din tupluri din M_C; $M_A \subset (M_C, M_C)$;
- O mulţime de specializări M_S, construită din tupluri din M_M şi M_A ; $M_S \subset (M_C, M_C)$.

Utilizînd aceste categorii, s-a introdus noţiunea de nivel de abstracţie:

Nivel de abstracţie Λ *al unui metamodel e o submulţime de clase astfel încît toate relaţiile care leagă nivelul cu alte clase ale metamodelului sunt orientate în aceeaşi direcţie şi de specializare sau generalizare (figura 1).*

Nivelele de abstracţie reprezintă un instrument util care contribuie mult la structurarea informaţiei în cadrul sistemului. Deşi o definiţie primitivă, nivelele de abstracţie se bazează pe definirea relaţiilor între clase şi din această cauză nu sunt suficiente pentru a descrie orice relaţie în cadrul unui proiect implementat, spre exemplu, specificarea regulilor pentru aplicarea într—un proiect elaborat într-un limbaj logic. Din această cauză trebuie de adoptat o altă abordare a conceptului de metodologie (caracterizată şi completată prin limbja şi metalimbaj de modelare) şi anume una care să specific relaţia exactă dintre metodologia de proiectare şi paradigmă de programare.

3. Metodologii legate de paradigme

Deobicei, limbajele propun metodologii care conţin un set de tehnici integrate cu scopul de a descrie sistemul.

În acest context, fiecare din tehnici are rolul său specific. De exemplu, tehnica *use-case analysis*, care e un instrument crucial în metodele OOSE, Booch Method şi UML, e o abordare metodică a problemei, cu ajutorul căreia putem să culegem informaţie despre elaborarea sistemului din specificaţiile sale. Ca rezultat al

aplicării acestei tehnici avem un set de scenarii care ulterior va fi utilizat pentru elaborarea claselor şi colaborărilor între ele.

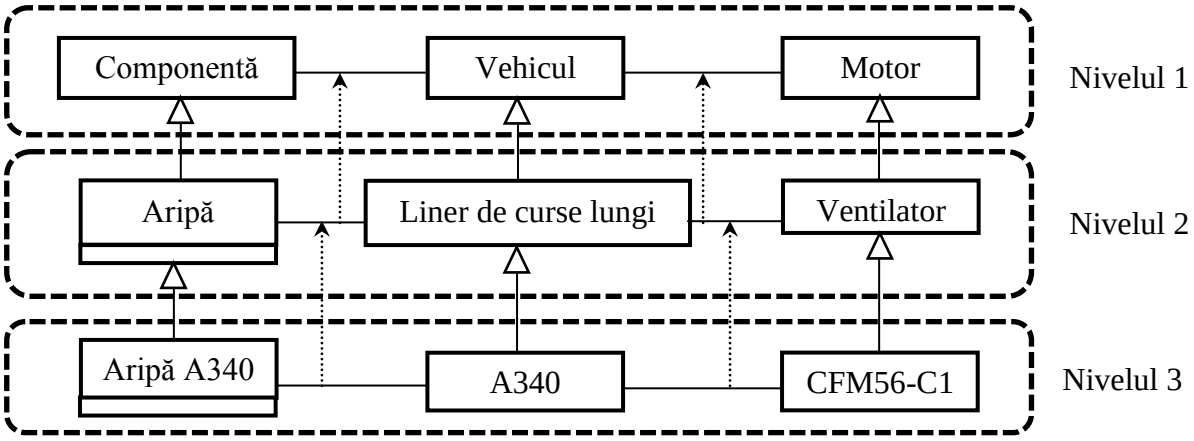

Fig. 1. Nivelele de abstracție

Majoritatea metodologiilor pot fi clasificate după paradigma pe care o prezintă [3]. Metoda Booch, OMT, OOA/OOD al lui Coad-Yourdon şi UML pot fi caracterizate drept metodologii obiect-orientate. La fel, SSADM şi SA/SD sunt abordări structurale care în mod firesc prezintă construcții şi tehnici utilizate în proiectarea imperativ-structurală.

Asocierea dintre paradigme şi metodologii nu este una aleatoare. Metodologiile structurale au fost introduse după depistarea greşelilor şi problemelor apărute la utilizarea limbajelor imperative. Ele aprobă o abordare anumită a procesului de dezvoltare a sistemului, concentrîndu-se pe fluxuri de date şi definirea pas cu pas a proceselor de sistem care vor fi dezvoltate utilizînd construcții procedurale.

În metode care se axează pe acțiune, tehnicile descriu fluxurile de date în sistem, utilizînd diagrame de fluxuri de date, structuri logice de date (definite într-o limbă vorbită) sau reprezentări grafice ale proceselor de schimb de date între componentele sistemului. În acest context, cuvintele cheie cu care operează aceste metodologii sunt de ordin imperativ: *secvențierea*, *selectarea* şi *iterația*.

Metodologiile OO conțin tehnici care descriu problema în termeni de abstracții care conțin starea sa [4]. Ele includ:

- modelele claselor principale şi asocierilor lor;
- obiectele care colaborează pentru a îndeplini o careva funcție;
- modele ale claselor cu dinamica semnificativă a stărilor lor.

În prezent, există puține metodologii pentru programare funcțională – deseori, metodologiile trebuiesc construite utilizînd abstracțiile proprii limbajelor funcționale. Brooks descrie o necorespondență dintre paradigme ca fiind un exemplu al *complexității accidentale*, care adaugă *complexității inițiale* a procesuslui de dezvoltare al produselor. Complexitatea esențială ține de tipul problemei şi familiaritatea cu domeniul problemei. Complexitatea accidentală poate fi evitată adaptînd o abordare consistentă în cadrul paradigmei de la modelarea cerințelor pînă la elaborarea soluției.

În acest context, doar adaptînd instrumentele de proiectare utilizate anterior, şi anume definirea pas cu pas a proceselor de sistem, design al fluxurilor de date şi o abordare modulară, nu utilizează beneficii caracteristice stilului şi limbajelor funcționale [5]. Această problemă a fost abordată de FAD, Yampa şi altele, însă fiecare din aceste metodologii propun tehnici specifice (puritate şi lipsa monadelor în cazul FAD şi programarea reactiv-funcțională pentru aplicații în timp real în cazul YAMPA) şi nu pot fi adaptate pentru dezvoltarea generală a softului pe principii funcționale.

4. Concluzii

Beneficiile pe care le are utilizarea unui limbaj sau metalimbaj de proiectare în cadrul unei metodologii de dezvoltare sunt incontestabile. În cazul în care se pledează pentru proiectarea unui produs utilizînd unul din exsistentele mijloace de modelare, abordarea pe care o propune metodologia de modelare se va reflecta direct în implementarea proiectului. E valabilă şi ordinea inversă, în situația în care echipa operează mai bine cu instrmente obiect-orientate, e mai rațional să se folosească un mijloc de proiectare care vizează procesul utilizînd concepte din paradigma respectivă. Cazul cu programarea funcțională este unul special – în prezent încă nu sunt metode şi tehnici suficient de universale pentru a acoperi toate necesitățile de proiectare a unui proiect utilizînd concepte aparținînd paradigmei funcționale.

Bibliografie

1. Schmidt, D. C. *Model-driven engineering* IEEE Computer, vol. 39, pp. 25–31, Februarie 2006,p. 148-149
2. Monperrus, M. et al. *A Definition of "Abstraction Level" for Metamodels*, 7th IEEE Workshop on Model-Based Development for Computer Based Systems, 2009
3. DeMarco, T. *Structured Analysis and system Specification*, Prentice-Hall, 1979
4. Pooley, R., Stevens, P. *Using UML: Software Engineering with Objects and Components*, Addison-Wesley, 1999
5. Russel, D. J. *FAD: A functional analysis and design metodhology*, Teză de doctorat , Universitatea din Kent, 2000