Comparative aspects about some object oriented methods

Prof.dr. Manole VELICANU, Catedra de Informatică Economică, ASE Bucureşti
Asist.drd. Ionel IACOB, Facultatea Informatică Managerială, URA Bucureşti

The fast evolution in the Information Technology domain meant, not only development of the hardware products, but also the elaboration of new methodologies for the development of Information Systems. Comparing to the traditional methods, object oriented approaching relies on the analysis phase, reducing this way a lot of the effort felt in the implementation stage. In practice, this thing leads to a better understanding of the real studied problem's tasks and also to building a reliable model, which is adaptable and can be easily modified afterwards.
Keywords: *information system, object oriented method, database, software engineering, design, analysis.*

Introducere

Din literatura de specialitate privind tehnologia orientată obiect, rezultă cinci *caracteristici* comune diferitelor metode de proiectare orientate obiect, pentru sistemele informatice:

- *Abstractizarea* (metoda). Acest concept include reguli şi instrucţiuni şi separă domeniul problemei în diferite abstractizări. În acest sens, există metode pentru identificarea stării, servicii şi interfeţe pentru încapsularea obiectelor şi strategii pentru abstractizarea similarităţilor în superclase abstracte.
- *Notaţii* şi verosimilitate reprezentaţională. Pentru reprezentarea conceptelor există notaţii referitoare la: agregare, moştenire, ierarhiile funcţionale, traficul mesajelor între obiecte, implementarea constrângerilor.
- *Validarea* (critica). Există reguli de transformare între: reprezentările alternative ale proiectării, validările între mesajele trimise şi protocoalele interfeţei, depăşirea punctelor slabe ale limbajului de implementare şi suportul pentru generarea de teste.
- *Reutilizarea.* Există concepte de identificare a claselor standard (oferite de sistem) din care se dezvoltă (de către analist) noi clase funcţionale, suport pentru diferite categorii de clase de obiecte şi caracteristicile lor de refolosire, independenţa limbajului pentru faza de proiectare logică.
- *Software*-ul. Fiecare metodă are implementări în mai multe produse software specializate, care funcţionează sub diferite sisteme de operare.

În continuare, vom realiza o sinteză comparativă privind principalele metode orientate obiect, luând în considerare, pentru fiecare, cele cinci caracteristici de mai sus.

1. OOD - Object-Oriented Development

METODA

Graddy Booch, într-o manieră exactă şi coerentă, a definit conceptele abordării orientate obiect şi a indicat folosirea următoarelor **etape** pentru analiza unui sistem orientat obiect:

- **Definirea problemei** de rezolvat. Problema este definită într-o descriere textuală concisă, iar apoi informaţiile despre obiectele şi operaţiile reprezentate în sistem pot fi obţinute din această descriere. Obiectele sunt reprezentate de substantive, iar operaţiile prin verbe.
- **Schiţarea unei strategii** de realizare a software-ului. Booch a precizat că dezvoltarea orientată obiect, aşa cum a descris-o el, nu este o metodă a ciclului complet de viaţă, ci a stagiilor de proiectare şi implementare.
- **Formalizarea strategiei**. *Ordinea* evenimentelor care duc la formalizarea strategiei este următoarea:

  • *Identificarea claselor şi obiectelor* implică, atât găsirea abstractizărilor cheie în spaţiul problemei de analizat, cât şi mecanismele importante care oferă un comportament dinamic mai multor asemenea obiecte. Aceste abstractizări cheie se găsesc studiind terminologia domeniului (problemei) de interes.

  • *Identificarea semanticii* implică stabili-

rea semnificației claselor şi obiectelor identificate în faza anterioară. Dezvoltatorul trebuie să privească obiectele din afară, să definească protocolul şi să investigheze felul în care fiecare obiect poate fi folosit de către celelalte obiecte.

• *Identificarea legăturilor* extinde activitatea anterioară pentru a include relațiile între clase şi obiecte precum şi modul de interacționare al acestora. Asocierile, precum moştenirile, instanțierile şi mesajele între clase sunt acum definite la fel ca şi semantica statică şi dinamică a mecanismelor inter-obiectuale.

• *Implementarea claselor* şi obiectelor implică examinarea acestora şi determinarea modului de implementare în limbajul de programare ales. Tot în această etapă se folosesc componentele de lucru, iar clasele şi obiectele sunt structurate în module.

Concluzii. Booch subliniază faptul că paşii de urmat integrează o dezvoltare iterativă şi incrementativă, prin rafinări ale vederilor complementare (logice şi fizice) ale unui sistem. În metoda sa, *Booch* subliniază diferența între o "vedere logică" a sistemului - în termeni de clase şi obiecte, şi o "vedere fizică" - module şi procese. De asemenea, el face diferența între *modelele statice* şi cele *dinamice* ale sistemului. Metoda propusă este totuşi una îndreptată mai mult către descrierile statice şi mai puțin către cele dinamice.

NOTAȚIA

Unul din punctele forte ale metodei lui *Booch* este abundența *notațiilor oferite:*

- există notații "diagramatice" (sub formă de simboluri) pentru producerea diagramelor de clasă (este principala diagramă şi expune o vedere statică a structurii de clasă), diagramelor de obiecte, diagramelor schimbărilor de stare, diagramelor de timp, diagramelor modulelor şi diagramelor proceselor;
- notațiile pentru modelarea claselor şi obiectelor folosesc legende şi simboluri variate (de exemplu, diferite tipuri de săgeți) pentru a transmite informații detaliate;
- un subset al notațiilor poate fi folosit în stadiile primare ale proiectării, iar detaliile se completează mai târziu;
- există o formă textuală pentru fiecare notație, constând în şabloane pentru documentarea fiecărei construcții principale;
- notațiile sunt definite într-un repertoar larg de simboluri, dar se pare că nu există totuşi o definire reală a sintaxei sau a semanticii statice, ci doar un text simplu care acompaniază simbolul.

CRITICA METODEI

Abordarea lui *Booch* este una fundamental pragmatică. Metoda de proiectare orientată obiect nu se dezvoltă niciodată cu adevărat într-un proces, fiind mai degrabă o colecție de tehnici, idei formalizate şi metodologii care pot fi folosite la dezvoltarea sistemelor orientate obiect. Ceea ce metoda nu explică este cum ar trebui proiectantul să decidă dacă continuarea descompunerii la un anumit stadiu este posibilă sau avantajoasă. Metoda nu oferă nici criterii de estimare a calității unei decompoziții în defavoarea alteia.

Booch utilizează o varietate de diagrame pentru diferite scopuri, dar în interiorul unora dintre diagrame el pare că încearcă să acopere prea mult din sistem.

Notațiile diagramelor lui Booch sacrifică detaliile (adâncimea), pentru lărgime. Aceasta poate să folosească în stadiile primare ale proiectării, dar trebuie să fie capabile de modificări pentru a mări gradul de detaliere. În parte, acest lucru este executat de şabloanele textuale, dar ele nu sunt structurate pentru a relata detalii într-un mod folositor pentru diagrame. De fapt, din anumite puncte de vedere, acestea mai sunt lipsite şi de informații esențiale pentru declararea unei bune structuri în interiorul aplicației.

SUPORT PENTRU REUTILIZARE

În privința reutilizării elementelor deja definite, în stadiul de proiectare al sistemului, Booch ia în considerare ambele variante: "proiectare pentru refolosire" şi "proiectare refolosind ceva existent".

Pentru prima variantă, din perspectiva procesului, *Booch* oferă indicații specifice pentru construirea componentelor refolosibile. Analistul de domeniu (din perspectiva orientat obiect) este responsabil cu identificarea, dezvoltarea, promovarea şi demonstrarea componentelor refolosibile în conjunctură cu proiectele.

Pentru a doua variantă, *Booch* e puțin mai

vag. El afirma că o activitate foarte importantă în orice dezvoltare este căutarea activă de componente software reutilizabile care sunt relevante pentru noul sistem. Dacă putem găsi o componentă relevantă pentru problema noastră, acea componentă devine o abstractizare primitivă cu care putem compune sistemul; în acest fel rămânem cu o problemă mai mică de rezolvat. În orice caz, detaliile sunt ascunse în exemplele oferite, şi nu se fac recomandări la modul general. *Booch* promovează şi folosirea generatoarelor de aplicaţii, acolo unde se poate.

PRODUSE SOFTWARE

Implementarea metodei OOD se regăseşte în produse software ale firmelor: Rational Rose (Unix, Windows), MetaEdit MetaCase Consulting (Windows), Unix Paradigm Plus Protosoft (Windows, Unix), System Architect Popkin Software (OS/2, Windows).

2. HOOD - Hierarchical Object-Oriented Design

METODA

HOOD a fost dezvoltată pentru Agenţia Spaţială Europeană ca metodă de proiectare şi porneşte de la un model iniţial care va fi rafinat în paşi succesivi pentru a fi implementat cu eforturi minime (este specifică proiectării şi implementării în limbajul ADA). Metoda presupune o descompunerea ierarhică top-down. Se pleacă de la o diagramă contextuală care expune toate interfeţele externe cu un singur proces central. Acest proces este apoi descompus în alte procese cu fluxuri de date şi fluxuri de control interacţionând între ele, cu verificări ale constanţei între niveluri. În acelaşi mod, scopul HOOD este să dezvolte proiectarea ca set de obiecte care împreună conferă funcţionalitatea programului.

Obiectele identificate pentru etapa de proiectare pot fi clasificate astfel: obiecte active (se pot executa în paralel), obiecte pasive (se pot executa doar secvenţial, la un anumit nivel), obiecte de mediu (reprezintă alte sisteme cu care interacţionează sistemul curent), obiecte clasă (sunt utilizate pentru a specifica un obiect al cărui tip nu este descris în întregime) obiecte ale nodului virtual (noduri într-un sistem distribuit).

Procesul principal în HOOD se numeşte *Basic Design Step* şi are ca scop identificarea unui obiect copil al unui obiect părinte dat şi a relaţiilor lor individuale cu alte obiecte existente, sau rafinarea unui obiect terminal până la nivelul codului. Acest proces se bazează pe identificarea obiectelor cu ajutorul tehnicilor de proiectare orientate obiect.

Fazele pentru parcurgerea metodei HOOD, care realizează un micro ciclu de viaţă pentru proiectare, sunt:

- *Definirea problemei*. Se declară contextul obiectului care urmează a fi creat, cu scopul de a organiza şi structura datele obţinute din faza de analiza a cerinţelor. Aceasta este o ocazie de a efectua o verificare completă a cerinţelor şi a transpunerii acestora în faza de proiectare, prin: *declaraţia problemei* - proiectantul declară problema în propoziţii corecte, care oferă o definiţie clară şi precisă a problemei, precum şi contextul sistemului care urmează a fi construit; *analizarea şi structurarea* datelor necesare - proiectantul adună şi analizează toate informaţiile relevante pentru problemă, incluzând mediul sistemului care urmează a fi construit.
- *Dezvoltarea strategiei soluţiei.* Schiţa soluţiei problemei expusă anterior se descrie în termeni de obiecte, la un nivel înalt de abstractizare, utilizând limbajul natural. Această descriere informativă prezintă etapa de proiectare prin prisma obiectelor lumii reale asociate cu acţiunile pe care le vor întreprinde. Se recomandă definirea a maximum zece propoziţii pentru schiţarea soluţiei de realizare.
- *Formalizarea strategiei.* Acum se definesc obiectele şi asocierile acestora şi se creează o diagramă a soluţiei propuse pentru a permite vizualizarea cu uşurinţă a conceptelor şi formalizarea lor în continuare. Exista cinci *subfaze* în formalizarea strategiei: identificarea obiectelor (prin extragerea substantivelor din descrierea soluţiei problemei şi păstrarea doar a celor relevante), identificarea operaţiilor (prin extragerea verbelor din enunţul problemei), gruparea obiectelor şi operaţiilor (prin ataşarea fiecărei operaţii la obiectul corespunzător), descrierile grafice ale obiectelor şi operaţiilor (prin utilizarea convenţiilor grafice HOOD, justificarea deciziilor de pro-

iectare (prin motivarea de către proiectant a deciziilor luate).
- *Formalizarea soluției*. Pentru formalizarea unei soluții se folosesc: definițiile formale ale interfețelor obiectelor, definițiile formale ale obiectelor şi structurilor operațiunilor de control, dezvoltarea unui model – numit Object Description Skeleton (ODS) – pentru fiecare obiect identificat, şi care conține interfețe şi structuri de control pentru obiecte şi operații.
NOTAȚIA
Principala reprezentare folosită pentru descrierea unui sistem informatic este *Diagrama obiect HOOD*, care oferă o vedere statică a structurii sistemului în ierarhia proiectării orientate obiect. Aceasta utilizează simboluri specifice pentru reprezentarea: claselor, obiectelor, operațiilor, legăturilor.
CRITICA METODEI
Metoda HOOD este puternic orientată către implementarea în limbajul ADA. Acest lucru este perfect pentru dezvoltătorii ADA, dar poate fi considerat limitat pentru orice alt mediu de programare.
Metoda oferă pasul proiectării de bază, dar nu ajută la descoperirea unei structuri potrivite pentru obiect. De fapt, HOOD oferă suport puternic pentru structuri de includere, dar nu şi pentru alte structuri, cum ar fi cele de moştenire.
Principala critică adusă metodei este lipsa de suport pentru unele dintre tehnicile (caracteristicile) orientate obiect disponibile. Suportul moştenirii este foarte redus, şi lipseşte orice notație care să o reprezinte grafic. Metoda este mai mult bazata pe obiecte, decât într-adevăr orientată obiect.
SUPORT PENTRU REUTILIZARE
Metoda HOOD nu ia în considerare în mod special reutilizarea. Unele principii - abstractizarea, încapsularea, modularizarea - ajută reutilizarea datorita aplicabilității la lumea reală şi la entitățile de date. Un obiect poate fi universal - deci reutilizabil într-o nouă configurație.
O clasă de obiecte trebuie creată diferit de proiectul principal, ca rădăcină: fie dintr-un proiect pre-existent, fie dintr-o bibliotecă de clase, fie a fost dezvoltat pentru a reprezenta un pachet generic ADA, fie că se dezvoltă special pentru proiectul în curs. Astfel, clasa are un scop general care poate fi folosit în mai multe părți ale proiectării. Aceasta este proiectarea de jos în sus (bottom-up).
PRODUSE SOFTWARE
Implementarea metodei HOOD se regăseşte în produse software ale firmelor: HOOD-SF Virtual Software Factory, Select Software Tools, Graph Talk Rank Xerox.

3. OMT - Object Modeling Technique

METODA
OMT oferă trei seturi de concepte (trei **viziuni** diferite asupra sistemului):
- *Modelul obiectual* - *descrie* structura statică a obiectelor dintr-un sistem şi legăturile dintre acestea; *conceptele* principale sunt: clase, atribute, operații, moşteniri, asocieri (relații), agregări; *rezultatul* este diagrama modelului.
- *Modelul dinamic* - *descrie* aspecte ale sistemului care se schimbă în timp şi este folosit pentru a specifica şi implementa aspectele de control ale sistemului; *conceptele* principale sunt: stări, sub/super stări, evenimente, acțiuni, activități; *rezultatul* este diagrama de stare.
- *Modelul funcțional* - *descrie* transformările valorilor datelor în interiorul sistemului; *conceptele* principale sunt: procese, stocuri de date, fluxuri de date, fluxuri de control, actori; *rezultatul* este diagrama de fluxului de date.
Fazele parcurgerii metodei OMT sunt:
▪ *Analiza de sistem* – se construieşte un model al situației din lumea reală, bazat pe declarația problemei şi cerințele utilizatorilor; *rezultatul*: declarația problemei, modelul obiectual, modelul dinamic, modelul funcțional.
▪ *Proiectarea sistemului* – se partiționează sistemul țintă în subsisteme, luând în considerare cunoştințele despre domeniul de interes şi arhitectura propusă pentru sistemul țintă (domeniul soluției); *rezultatul* este Documentul de Proiectare al Sistemului (arhitectura de bază a sistemului şi deciziile strategice).
▪ *Proiectarea obiectelor* - se realizează proiectarea bazată pe modelul analitic îmbogățit cu detalii de implementare*; rezultatul* este dat de modelele detaliate: obiectual, dinamic,

funcţional.

- *Implementarea* – se defineşte transpunerea rezultatelor etapei de proiectare în limbaje de programare, cu un accent deosebit pe posibilitatea de depanare, flexibilitate şi extensibilitate.

NOTAŢIA

Principala diagramă folosita pentru descrierea structurii unui sistem se defineşte în cadrul *Modelului Obiectual*, cea care oferă un model al structurii de clasa specificat în proiectarea orientată obiect. Se utilizează *simboluri* specifice pentru reprezentarea în diagrame a: claselor, atributelor, operaţiilor, instanţelor, asocierilor, caracteristicilor obiectelor (moştenirea, agregarea).

CRITICA METODEI

OMT este una dintre cele mai dezvoltate metode de proiectare orientată obiect, atât din punct de vedere al notaţiilor folosite cât şi din punct de vedere al proceselor recomandate pentru dezvoltarea unui sistem orientat obiect.

Metoda încearcă să arate cum se folosesc conceptele orientate obiect de-a lungul întregului ciclu de viaţă al dezvoltării sistemelor informatice.

Conceptul folosirii a trei viziuni pentru reprezentarea unui sistem în *OMT* este foarte puternic şi poate fi considerat de asemenea foarte complex.

Nu este foarte clar dacă viziunile trebuie dezvoltate independent una de cealaltă sau dacă informaţiile dintr-un model trebuie folosite pentru a influenţa construcţia altuia. Totuşi, pare logic ca noţiunile diferitelor viziuni, să fie definite corect şi corelate între ele, altminteri putând apărea multe confuzii în faza de proiectare a dezvoltării, unde trebuie implementate modelele.

Nu există încercări serioase de abordare a reutilizării sistematice a software-ului sau componentelor.

Managementul procesului de dezvoltare, inclusiv stabilirea unor unităţi de măsură potrivite pentru determinarea progresului şi calităţii, este aproape total neglijat.

Din punct de vedere al abordării tehnice a dezvoltării sistemului orientat obiect, au stabilite clar standardele tehnologiei moderne.

Unii cercetători (Rumbaugh) recomandă includerea cazurilor de utilizare ale lui I.Jacobson în metoda OMT.

Conform unui sondaj al OMG (Object Management Group), OMT este în prezent cea mai populară metodă standardizată de dezvoltare orientată obiect. Unul dintre secretele succesului acesteia poate fi varietatea de instrumente ajutătoare disponibile.

SUPORT PENTRU REUTILIZARE

La aplicarea metodei se ia în considerare numai refolosirea codului. Se poate refolosi un modul dintr-o proiectare anterioară, dacă este posibil, dar trebuie evitat să se forţeze compatibilitatea. Cel mai uşor este atunci când o parte a domeniului problemei corespunde unei probleme anterioare. Dacă problema nouă este asemănătoare problemei anterioare, dar diferită, s-ar putea că proiectarea originală să fie extinsă pentru a cuprinde ambele probleme.

A plănui ceva pentru reutilizări viitoare cere multă putere de prevedere şi reprezintă o investiţie. Este puţin probabil că o clasă izolată va fi folosită pentru proiecte multiple. Proiectanţii vor refolosi mai degrabă nişte subsisteme bine studiate, cum ar fi tipuri de date abstracte, pachete grafice şi biblioteci de analize numerice.

PRODUSE SOFTWARE

Implementarea metodei OMT se regăseşte în produse software ale firmelor: SelectOMT (Windows), StP/OMT Interactive Development Environments (AIX), Excelerator II Intersolv (Windows), MetaEdit MetaCase Consulting (Windows).

4. RDD – Responsibility Driven Design

METODA

În metoda RDD, un model se dezvoltă din specificaţiile cerinţelor, prin extragerea substantivelor şi verbelor. Aceasta oferă o bază pentru implementarea propriu-zisă. Toate conceptele esenţiale ale orientării obiect sunt acoperite. Metoda se mai numeşte şi CRC – Class Responsability Collaboration.

În RDD, pentru fiecare clasă se defineşte diferit *responsabilitatea*, care specifica rolul obiectelor şi acţiunile lor. Pentru a îndeplini această responsabilitate, clasele trebuie să colaboreze între ele. *Colaborările* sunt definite

în aşa fel încât să arate cum vor interacţiona obiectele. Responsabilităţile sunt grupate mai departe în *contracte* care definesc un set de cerinţe pe care obiectele claselor le pot îndeplini. Contractele sunt mai departe rafinate în *protocoale*, care expun semnătura specifică fiecărei operaţii. *Subsistemele* sunt introduse pentru a grupa un număr de clase şi subsisteme de nivel inferior, cu scopul de a abstractiza o anumită funcţionalitate. Subsistemele au, de asemenea, contracte care trebuie suportate de anumite clase din subsistem.

Metoda RDD presupune parcurgerea a două **faze** principale, după cum urmează:

- explorarea care constă în definirea următoarelor concepte: clase, responsabilitatea claselor, colaborările dintre clase.

- analiză care constă în definirea următoarelor structuri: ierarhiile de clase, subsistemele posibile, protocoalele pentru fiecare clasă.

NOTAŢIA

Principala diagramă folosită la descrierea structurii sistemului este *Graficul Colaborărilor,* care reprezintă subsistemele, clasele, contractele şi colaborările în sistem. Pentru reprezentarea în diagrame, se folosesc *simboluri* care se referă la: clase, contracte, subsisteme, colaborări.

CRITICA METODEI

Metoda RDD tinde să folosească *tehnici obişnuite* şi instrucţiuni pentru dezvoltarea unei etape de proiectare corespunzătoare, bazându-se mult pe abilitatea proiectantului de a descoperi clasele implicate şi proprietăţile acestora.

O parte importantă a strategiei de proiectare este *analiza textuală a specificaţiilor cerinţelor* pentru identificarea claselor şi responsabilităţilor.

Se folosesc carduri clasă-responsabilitate-colaborare (CRC – carduri index) care sunt ideale pentru captarea specificaţiilor claselor şi subsistemelor, deoarece sunt compacte, uşor de manipulat şi uşor de actualizat. De asemenea, pot fi uşor aranjate şi rearanjate pe ecran pentru a obţine o nouă perspectivă. Acest tip de simplitate poate fi considerat şi avantaj şi dezavantaj, deoarece face metoda de proiectare mult mai uşor de manevrat din punct de vedere intelectual şi notaţional, dar există şi o puternică dependenţă faţă de claritatea şi precizia documentului cu specificaţiile cerinţelor.

RDD ia în considerare numai activitatea de proiectare, fără a se da nici o atenţie analizei domeniului, respectiv problemei sau captării cerinţelor.

Conceptul de clasă este central – toate celelalte concepte identificându-se în metodă doar pentru a descrie clasele şi interdependenţele lor.

Activitatea de proiectare produce ceea ce este de fapt un singur model într-o singură notaţie. Nu există alte modele, cum ar fi, de exemplu, modele ale comportamentului dinamic. Notaţia nu este suficient de expresivă pentru a reliefa în mod adecvat proiectarea.

SUPORT PENTRU REUTILIZARE

În metodă se ia în considerare atât "proiectarea pentru reutilizare ulterioară" cât şi "proiectarea reutilizând ceva existent".

Atunci când se identifică clase şi subsisteme, metoda încurajează folosirea componentelor implementate anterior în proiectul actual.

Dacă o clasă sau subsistem implementate anterior pot îndeplini responsabilităţile cerute de sistem, acestea trebuiesc refolosite.

De asemenea, în identificarea claselor candidate, proiectantul trebuie să identifice candidatele la superclase abstracte, care pot fi făcute reutilizabile urmând instrucţiunile.

PRODUSE SOFTWARE

Implementarea metodei RDD se regăseşte în produse software ale firmelor: Excelerator II Intersolv (Windows), HomeSuite Hattearas Software (Windows), TurboCase StructSoft (Mac).

5. OOA - Object-Oriented Analysis

METODA

Metoda OOA, concepută de Coad şi Zourdon, foloseşte principiile de bază ale structurării şi le reuneşte într-o viziune orientată obiect. Ea presupune separarea domeniului lumii reale de funcţiile pe care trebuie să le îndeplinească sistemul pentru utilizatorii săi. În acest mod, modificările în cerinţele funcţionale nu vor avea efect asupra altor părţi ale sistemului care nu au legătură cu cerinţele respective.

Metoda OOD presupune parcurgerea a cinci

paşi:

- **Identificare claselor şi obiectelor** - specifică cum ar trebui descoperite clasele şi obiectele; se începe cu domeniul aplicaţiei şi se identifica clasele şi obiectele care formează baza întregii aplicaţii şi, din această perspectivă, se analizează responsabilităţile sistemului în acest domeniu.
- **Identificarea structurilor** - se defineşte structura *generalizare – specializare*, care captează ierarhia claselor identificate; se identifică structura *parte – întreg*, care se foloseşte pentru a arata cum un obiect este parte a altui obiect, şi cum obiectele sunt compuse în categorii mai largi.
- **Definirea subiectelor** - se face prin partiţionarea *Modelului Clasă & Obiect* în unităţi mai largi; subiectele sunt grupuri ale modelului; structurile identificate anterior pot fi folosite.
- **Definirea atributelor** - se identifică informaţiile şi asocierile pentru fiecare instanţă; se identifică atributele necesare pentru a caracteriza fiecare obiect; se plasează la nivelul corespunzător în ierarhia moştenirii.
- **Definirea serviciilor** – se definesc operaţiile claselor; se identifică starea obiectelor şi se definesc serviciile pentru accesarea şi actualizarea acelei stări.

Rezultatul este un model al Domeniului Problemei structurat pe cinci *nivele*, după cum urmează: nivelul subiectului, nivelul clasă şi obiect, nivelul structură, nivelul atribut, nivelul servicii. Identificarea elementelor fiecărui nivel constituie o activitate în interiorul metodei. Ordinea abstractizărilor nu este importantă, deşi în general este mai uşor să se treacă de la nivelurile superioare de abstractizare la cele inferioare.

NOTAŢIA

Principala diagramă folosită pentru descrierea unui sistem este *Modelul OOA,* care oferă o vedere statică a structurii clasei într-o proiectare orientată obiect. Pentru reprezentarea în diagrame, se folosesc *simboluri* care se referă la: clase, obiecte, atribute, servicii, structuri (moştenirea, parte-întreg), conexiuni, instanţe, mesaje.

CRITICA METODEI

Metoda OOA lasă impresia că a fost adaptată la repezeală din alte metode clasice (structurate) dar la timp pentru a sări în "vagonul orientat obiect".

Porţiuni din expunerea metodei sunt excesiv de detaliate pentru cititorii orientaţi către management, în timp ce alte secţiuni sunt mult prea generale pentru cei din proiectare şi implementare.

Componentele metodei nu sunt prezentate suficient de detaliat pentru a fi puse direct în practică, dar oricum, abordarea lor din punct de vedere al analizei şi proiectării orientate obiect, este una uşor de înţeles iar metoda este explicată la un nivel care evită unele dintre complexităţile orientării obiect formale.

Aşa cum este ea definită, metoda oferă o bună introducere în orientarea obiect dar are şi suficiente defecte pentru a fi evitată în cazul sistemelor mari sau foarte complexe.

SUPORT PENTRU REUTILIZARE

Metoda OOA ia în considerare numai "proiectarea utilizând ceva existent". Ea se concentrează exclusiv pe componentele domeniului problemei: o analiză mai eficientă necesita folosirea construcţiilor domeniilor problemei, atât pentru folosirea curentă, cât şi pentru cea viitoare. O singură componenta a metodei lor, aceea a Domeniului Problemei, este selectată pentru o potenţială refolosire.

PRODUSE SOFTWARE

Implementarea metodei OOA se regăseşte în produse software ale firmelor: ObjectTool (Windows), Object International (Windows), HP (Windows), ObjectModeler Iconix Software Engeneering (Mac), OOTher Roman (Windows).

Bibliografie

1. I.Lungu, Gh.Sabău, M.Velicanu ş.a. – Sisteme informatice – analiză, proiectare, implementare, Ed.Economică, 2003.
2. M.Velicanu, I.Lungu, M.Muntean – Sisteme de baze de date – teorie şi practică, Ed. Petrion, 2003.
3. G.Booch – Object Oriented Analysis and Design, Ed. Addison-Wesley, 1994.
4. R.S.Pressman – Software engineering – a practitioner's approach, Ed. McGraw-Hill, 2000.
5. J.Rumbaugh – Object Oriented Modeling and Design, Ed. Prentice Hall, 1994.
6. P.Codd, E.Yourdon – Object oriented analysis, Ed. Prentice Hall, 1991.
7. J.Hoffer, J.George – Modern systems ana-lysis and design, Ed. Addison-Wesley, 1999.
8. I.Jacobson, G.Booch – The Unified Software Development Process, Ed. Addison-Wesley, 1999.
9. J.Norman – Object oriented systems analysis and design, Ed. Prentice Hall, 1996.