Software development methodologies

Lect. dr. Paul POCATILU
Catedra de Informatică Economică
A.S.E., Bucureşti

Developing project portfolio management applications and other applications requires the use of a software development methodology and specific tools. This paper presents several methodologies for software development that can be used for project portfolio applications as well.
***Keywords**: project portfolio management, software life cycle, software development, methodologies*

Introducere

Evoluția societății informaționale se realizează prin intermediul soluțiilor software şi hardware elaborate de firmele din domeniul tehnologiei informației (IT). În societatea actuală, majoritatea companiilor IT îşi desfăşoară activitățile prin intermediul proiectelor, la un moment dat derulându-se mai multe proiecte în paralel. Managementul proiectelor IT desfăşurate în cadrul firmelor este esențial pentru succesul aplicațiilor realizate. Numărul şi complexitatea proiectelor care se derulează variază de la o firmă la alta, de la o perioadă la alta. Proiectele care se derulează la un moment dat într-o firmă partajează aceleaşi resurse. Prin organizarea proiectelor în cadrul unui portofoliu de proiecte se urmăreşte obținerea unei imagini de ansamblu asupra tuturor proiectelor companiei în vederea unui management centralizat al acestora. Se are în vedere stabilirea unei metodologii care să permită realizarea de aplicații pentru managementul portofoliilor proiectelor IT care să țină cont de necesitățile actuale ale managerilor şi de modul în care se desfăşoară derularea proiectelor. Astfel, se are în vedere distribuirea în spațiu a proiectelor şi necesitatea de a avea informații în orice moment şi de oriunde despre proiectele desfăşurate. Ținând cont de aceste lucruri, metodologia are în vedere dezvoltarea de *aplicații distribuite* precum şi *aplicații mobile*.

În continuare sunt prezentate caracteristicile principalelor metodologii utilizate în dezvoltarea software

Tehnici şi metode utilizate în dezvoltarea software

Tehnicile de dezvoltare utilizate în realizarea produsului au în vedere etapele de analiză şi proiectare a acestuia. Există mai multe metode de analiză şi proiectare, clasificate după modul în care este realizat proiectul aplicației. Astfel, există metode care au la bază:

- structura funcțională; în cadrul acestor metode sistemul este împărțit în subsisteme pe baza funcționalității componentelor;
- structura datelor; proiectarea sistemului se bazează pe datele care sunt prelucrate;
- fluxul datelor; în proiectare se porneşte de la fluxurile datelor din cadrul sistemului.

Metodele care au la bază structura funcțională a produsului sunt:

- metoda Top-Down; are la bază principiul modularității şi presupune descompunerea sistemului de sus în jos până la obținerea de module elementare;
- metoda Bottom-Up; la baza metodei stă principiul agregării şi presupune identificarea de jos în sus a componentelor sistemului;
- metoda HIPO; metoda se bazează pe descrierea intrărilor, prelucrărilor şi ieşirilor într-o manieră ierarhică; sunt construite diagramele de conținut H şi diagramele IPO, pentru intrări/prelucrări/ieşiri;
- metoda de analiză şi proiectare structurală – SADT; la baza metodei stă descompunerea descendentă pe baze funcționale a sistemului; metoda presupune elaborarea a două modele: modelul activităților şi modelul datelor.

Metodele care pornesc de la structura datelor sunt:

- metoda de realizare a sistemelor – LCS şi metoda de realizare a programelor – LCP; au la bază structurarea sistemului în funcție de

datele de ieşire;
- metoda Jackson; sistemul este structurat pornind de la analiza datelor problemei; se creează o structură ierarhică a sistemului.

Metodele care au în vedere fluxul datelor sunt:
- metoda lui DeMarco; este o metodă de analiză structurată;
- metoda lui Yourdon şi Constantine; se obţine diagrama de flux a datelor (DFD) şi pe baza acesteia se realizează o descompunere funcţională a sistemului;
- metoda lui Myers; se mai numeşte şi proiectarea compozită; obiectivul metodei este acela de a minimiza relaţiile din interiorul modulelor şi de a maximiza relaţiile dintre module.

Metodologii orientate obiect

Metodele de analiză şi proiectare orientate obiect pun accentul în primul rând pe reutilizare. Principalele metode de analiză şi proiectare orientate obiect sunt OOA (Object Oriented Analysis), OOD (Object Oriented Design), OOSE (Object Oriented Software Engineering), OMT (Object Modeling Technique) şi RUP (Rational Unified Process).

OOA este o metodă de analiză orientată obiect dezvoltată de Shlaer & Mellor care se bazează pe o multitudine de reguli, analistul având destul de puţine posibilităţi de reprezentare a problemei sau a lumii reale. Analistul partiţionează sistemul în domenii, construindu-se diagrame ale domeniile. Fiecare domeniu este analizat separat, acestea fiind interconectate în timpul proiectării şi implementării. Modelele cu care lucrează metoda sunt:
- modelul informaţional; sunt descrise entităţile conceptuale şi relaţiile dintre acestea;
- modelele stărilor; conţine ciclurile de viaţă a obiectelor şi relaţiile dintre obiecte;
- modelul de comunicaţie a obiectelor; este redată comunicaţia asincronă dintre obiectele din sistem;
- modelul de acces al obiectelor; descrie comunicaţia sincronă dintre obiectele sistemului.

Metoda de analiză orientată obiect ia în considerare şi posibilitatea că proiectarea orientată obiect nu poate fi aplicată pentru problemă şi lasă la latitudinea analistului sau proiectantului această alegere. OOA lucrează cu diagrama claselor, diagrama de structură a claselor, diagrama dependenţelor şi diagrama moştenirii [SHLA92].

OOD a fost descrisă pentru prima dată de către G. Booch în [BOOC91]. Metoda subliniază rolul esenţial jucat în proiectarea orientată-obiect de aspectul iterativ al dezvoltării şi importanţa creativităţii dezvoltatorului. Metoda este mai mult decât un set de reguli euristice şi sfaturi referitoare la procesul realizării unei aplicaţii. Deşi nu sunt prezentate reguli stricte şi succesiune riguroasă în timp, ordinea operaţiilor care trebuie efectuate în OOD este următoarea:
- identificarea claselor şi obiectelor la un anumit nivel de abstractizare;
- identificarea semanticii claselor şi obiectelor;
- identificarea relaţiilor între clase şi obiecte;
- implementarea claselor şi obiectelor.

Metoda se concentrează asupra fazei de proiectare, pentru care oferă un set bogat de concepte şi notaţii. Prezentarea modelului de dezvoltare din diferite perspective permite ilustrarea clară a unor aspecte diferite ale modelului. OOD separă structura logică de cea fizică. Vederea logică conţine structura claselor, în timp ce vederea fizică se referă la structura modulelor şi a proceselor. Diagramele detaliate pentru module şi procese constituie una din caracteristicile OOD. Pe lângă diagramele statice, Booch utilizează şi două diagrame dinamice. Prima este o diagramă de tranziţie a stărilor iar a doua, o diagramă de timp care descrie succesiunea evenimentelor între obiecte.

OMT acoperă fazele de analiză, proiectare şi implementare. Spre deosebire de OOD, OMT se concentrează asupra analizei. Metoda, concepută de Rumbaugh, este descrisă în [MIHA98] şi constă din patru etape:
- analiza; constă în definirea domeniului aplicaţiei fără a lua în considerare aspecte legate de implementare; rezultatele etapei sunt modelele obiectual, dinamic şi funcţional;
- proiectarea sistemului; modelele obţinute în faza de analiză sunt rafinate, extinse şi detaliate astfel încât pe baza lor să se implementeze; ca rezultat al fazei de proiectare se obţine arhitectura sistemului;
- proiectarea orientată obiect; în această etapă sunt proiectaţi algoritmii, asocierile şi se creează ierarhiile de clase; rezultatele acestei eta-

pe sunt cele trei modele detaliate;
- implementarea; pe baza modelelor detaliate este scris codul aplicației utilizând un limbaj de programare orientat obiect.

Inițial sunt dezvoltate trei modele ale sistemului, care ulterior sunt rafinate în toate etapele ciclului de viață. Aceste modele sunt:
- modelul obiect (static) care descrie structura statică a sistemului luând în considerare clasele şi relațiile dintre ele;
- modelul dinamic, construit pentru a surprinde aspectele temporale ale modelului obiect;
- modelul funcțional care descrie în principiu, în termenii operațiilor dintre obiecte, modul în care plecând de la informațiile inițiale sunt obținute informațiile finale.

OOSE (Ivar Jacobson) este o metodologie orientată obiect bazată pe idei noi privind procesul de realizare a software-ului. Etapele acestei metodologii sunt [JACO96]:
- analiza; sunt identificate clasele sistemului; din etapa de analiză rezultă modelul cerințelor şi modelul de analiză;
- construcția; etapa constă din proiectare şi implementare; clasele obținute în etapa de analiză sunt rafinate; sistemul este adaptat mediului real în care acesta operează, fiind luate în considerare limbajul de programare, bazele de date cu care acesta lucrează; rezultatele etapei sunt modelul de proiectare şi modelul de implementare;
- testarea; fiecare clasă sau grup de clase este testat în parte atât din punct de vedere funcțional cât şi structural; instanțe ale claselor sunt integrate continuu pe parcursul realizării sistemului şi testate; după integrarea întregului sistem, acesta este testat; pentru testarea de integrare şi testarea de sistem sunt utilizate cazurile de utilizare;
- mentenanța; după acceptarea sistemului de către beneficiar, acesta intră în exploatare curentă; în măsura în care apar erori pe parcursul utilizării, acestea sunt corectate.

Metoda utilizează cinci modele corespunzătoare etapelor de dezvoltare: modelul cerințelor, de analiză, de proiectare, de implementare şi de testare. Fiecare model ia în considerare o anumită parte sau anumite aspecte ale sistemului care se realizează.

Un concept nou al metodei este cel al cazurilor de utilizare (*Use Cases*), concept care face parte din modelul cerințelor, acesta făcând posibilă legătura dintre cerințe, dezvoltare, testare şi acceptare de către utilizatorul final [MIHA98], [BODE01].

Metoda *RUP* are la bază metodele OOD (Booch), OMT (Rumbaugh) şi OOSE (Jacobson). Pe lângă aceste metode, au mai fost folosite şi FUSION (Coleman), OOA (Shlaer & Mellor) şi alte metode. Metoda unificată se bazează pe cazurile de utilizare, este centrată pe arhitectură şi este interactivă şi incrementală. *UML* stă la baza reprezentărilor grafice utilizate în această metodă. Principiile reunificării [RUMB96], sunt:
- simplitatea – metoda trebuie să folosească un număr redus de concepte;
- aplicabilitatea – metoda trebuie să poată fi aplicabilă la construirea unui număr cât mai mare de sisteme diferite;
- utilitatea – metoda trebuie să se concentreze asupra acelor elemente care sunt semnificative pentru un sistem practic şi pentru tehnicile de ingineria programării;
- atenția specială acordată problemelor frecvent întâlnite – problemele frecvent întâlnite trebuie să fie simplu de modelat iar problemele limită pot să necesite o complexitate oarecare, măsurată prin frecvența de utilizare;
- auto-consistența – acelaşi concept şi simbol trebuie să fie aplicat în aceeaşi manieră în cadrul metodei;
- ortogonalitatea – conceptele independente trebuie să fie modelate independent;
- structurarea pe niveluri – conceptele avansate trebuie tratate ca extensii ale conceptelor de bază ale metodei;
- stabilitatea – să adopte concepte şi simboluri care sunt larg cunoscute şi folosite;
- posibilitatea de tipărire – diagramele folosite trebuie să poată fi uşor desenate cu mâna sau cu editoarele grafice;
- extensibilitatea – atât utilizatorii cât şi constructorii trebuie să aibă un anume grad de libertate care să le permită să extindă şi să adapteze metoda.

Metodologii light

S-au dezvoltat o serie de metodologii de analiză şi proiectare pentru proiecte de dimensiuni

reduse, precum: Adaptive Software Development (ASD), Extreme Programming (XP), SCRUM şi Crystal Clear. Majoritatea acestor metodologii sunt orientate obiect.

Metodologia ASD se bazează pe teoria sistemelor complexe adaptive, utilizând trei concepte importante pentru a descrie lumea externă: agenţi, medii şi rezultate.

La baza metodologiei Extreme Programming (XP) stă satisfacţia clientului şi lucrul în echipă; metoda se utilizează pentru proiecte la care lucrează între 2 şi 10 oameni; în echipă sunt incluşi şi managerii şi clienţii; o cerinţă importantă este testabilitatea şi se are în vedere posibilitatea automatizării testelor de modul şi funcţionale; obiectivul principal este livrarea de software de calitate la timp; în privinţa testării se scriu mai întâi cazurile de test şi după aceea programele care urmează a fi testate.

Obiectivul metodologie SCRUM ei este de a furniza software de calitate într-o serie de trei până la opt intervale de timp care durează cel mult o lună; etapele metodei, stabilirea cerinţelor, analiza, proiectarea, evoluţia şi livrarea sunt grupate într-un stagiu; aceste stagii de derulează până la finalizarea produsului software.

Metodologia Crystal Clear se adresează proiectelor la care lucrează echipe între 4 şi 6 oameni; se bazează pe elementele de succes ale proiectelor precedente, iar accentul este pus pe comunicaţia continuă în cadrul echipei.

Toate aceste metodologii pun un accent important pe calitate şi pe testarea rezultatelor obţinute. [DIAS00]

Concluzii

Utilizarea instrumentelor CASE în analiza şi proiectarea aplicaţiilor software pentru anumite tehnici şi metode de dezvoltare conduce la:

- creşterea productivităţii în activitatea de analiză şi proiectare; având la dispoziţie o multitudine de posibilităţi de descriere grafică a modelelor şi facilităţi de verificare a consistenţei acestora, efortul necesar analizei şi proiectării se reduce;
- obţinerea de documente unitare în fiecare etapă de dezvoltare; după fiecare etapă sunt generate automat documente specifice pentru descrierea modelelor elaborate şi a relaţiilor dintre acestea;
- posibilitatea generării de cod sursă pentru modelele realizate; modelele detaliate obţinute permit generarea de cod în diverse limbaje de programare (C++, Java, VB); sunt create clasele şi prototipurile metodelor, urmând ca algoritmii şi funcţionalitatea acestora să fie completate de către programatori.

La ora actuală există o multitudine de instrumente CASE care suportă atât metode de analiză şi proiectare structurate, cât şi metode orientate obiect.

Bibliografie

[BODE01] Bodea, Constanţa-Nicoleta, Sabău, Gheorghe, Posdarie, Elena – *Sisteme Informatice Economice. Analiză şi proiectare orientate obiect utilizând UML*, Editura Inforec, Bucureşti, 2001

[BOOC91] Booch, Grady, *Object Oriented Design with Applications*, The Benjamin/Cummings Publishing Company, Inc., 1991

[DIAS00] Dias, Paulo – *Lightweight methodologies*, INESC, 2000

[JACO96] Jacobson, Ivar et al – *Object-Oriented Software Engineering* – A Use Case Driven Approach, Addison-Wesley, 1996

[MIHA98] Mihalca, Rodica, Fabian, Csaba, Uţă, Adina, Simion, Felix – *Analiză şi proiectare orientate obiect – Instrumente de tip CASE*, Editura Societatea Autonomă de Informatică, Bucureşti 1998

[MIHA02] Mihalca, Rodica – *Programe aplicative*, Biblioteca Virtuală ASE, Bucureşti, 2002

[POCA04] Pocatilu, Paul – *Project Portfolio Management Applications*, Revista Economy Informatics, No. 1, 2004, pp.73-76

[RUMB96] Rumbaugh, James E. – *To Form a More Perfect Union: Unifying the OMT and Booch Methods*, JOOP, Vol. 8, No. 8, January 1996, pp. 14-18, 65

[SHLA92] Shlaer, Sally – *A Comparison of OOA and OMT*, Project Technology, 1992