Project Portfolio Management Applications Testing

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

Many IT companies are running project simultaneously. In order to achieve the best results, they have to group to the project in portfolios, and to use specific software that helps to manage them. Project portfolio management applications have a high degree of complexity and they are very important for the companies that are using it. This paper focuses on some characteristics of the testing process for project portfolio management applications.
Keywords*: software testing, projects, project portfolio management, application development.*

Aplicații pentru managementul portofoliilor proiectelor (MPP)

Proiectele IT sunt caracterizate prin elemente variate şi managementul acestor proiecte este influențat de o serie de factori care trebuie să fie identificați prin această cercetare. Pentru a dezvoltare un sistem software pentru MPP, fazele de analiză şi proiectare trebuie parcurse, după o documentare riguroasă în domeniul cercetat

Aplicațiile pentru MPP implementează componente pentru (figura 1):

- introducerea şi actualizarea datelor despre proiecte şi criteriile de selecție
- selecția proiectelor
- optimizarea portofoliului de proiecte
- situații şi rapoarte
- urmărirea proiectelor
- imprimarea
- culegerea de date şi calcularea de metrici.

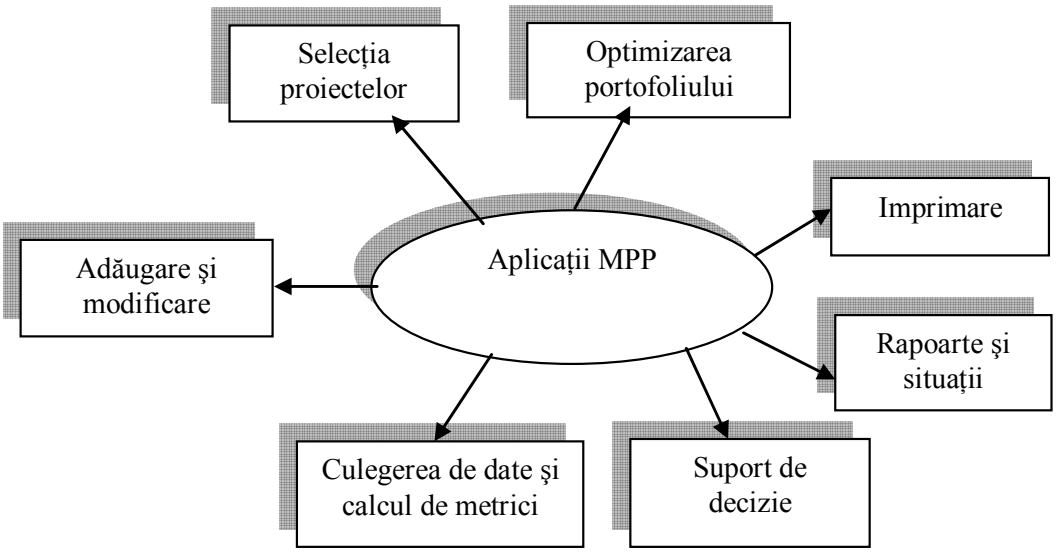

Fig.1 Funcțiile aplicațiilor de pentru MPP

Toate informațiile despre proiecte şi programe trebuie stocate în baze de date. Despre fiecare proiect sau program se înregistrează următoarele caracteristici:

- denumirea proiectului
- date despre proprietarul proiectului
- date despre managerul proiectului
- date despre organizație
- tipul proiectului
- data de început şi data de sfârşit a proiectului
- costurile de investiție
- profitul aşteptat
- riscurile implicate
- relațiile existente.

Aplicațiile pentru MPP sunt dezvoltate pentru diferite platforme având arhitecturi specifice, tabelul 1.

Tabelul 1 Tipuri de aplicaţii pentru MPP

Tipul aplicaţiei	Interfaţa utilizator	Necesar de memorie	Putere de procesare	Posibilităţi de acces	Disponibilitate
Desktop	Complexă	Înalt	Înaltă	Fix	Limitată
Mobilă	Limitată	Scăzut	Scăzută	Oriunde	Oricând
Bazată pe Web	Specifică Web	Mediu	Scăzută-medie	Oriunde	Oricând

Din tabelul 1, se observă că principalele avantaje ale aplicaţiilor mobile şi bazate pe Web pentru MPP sunt posibilitatea de acces de oriunde şi oricând, însă se remarcă limitările interfeţei cu utilizatorul. Un dezavantaj al aplicaţiilor bazate pe îl constituie necesitatea existenţei conexiunii la Internet, defecţiunile apărute în acest caz conducând la nefuncţionalitatea aplicaţiei.

Testarea software

Testarea reprezintă o etapă importantă în procesul de realizare a produselor software. În [PETE00], [VLIE00] se specifică faptul că ponderea cheltuielilor cu testarea reprezintă între 30% şi 50% din totalul cheltuielilor pentru dezvoltarea unei aplicaţii software.

Tehnicile şi metodele moderne de elaborare a produselor software acordă o importanţă deosebită efortului de înlăturare a erorilor de analiză, proiectare şi programare prin folosirea unor mijloace evoluate de testare. Există numeroase modalităţi de a realiza testarea aplicaţiilor software, dar în toate cazurile se porneşte de la obiectivele specifice şi de la resursele disponibile.

Testarea se concentrează atât asupra logicii interne a programului, avându-se în vedere ca anumite elemente ale acestuia să fie parcurse, cât şi pe funcţionalitatea externă a sa, pe baza specificaţiilor. Se compară rezultatele efective obţinute după rularea programului cu seturi de date de test cu rezultatele scontate pe baza specificaţiilor.

Testarea sau analiza statică are ca scop examinarea aplicaţiilor software fără a fi executate şi cuprinde activităţi precum inspecţiile, execuţia simbolică şi verificarea. Aceste activităţi fac parte din categoria evaluările tehnice [MYER04].

Testarea dinamică presupune examinarea aplicaţiilor software în scopul generării datelor de test cu care acestea vor fi executate şi rularea aplicaţiilor cu seturile de date de test obţinute. Se observă că spre deosebire de testarea statică, testarea dinamică presupune execuţia aplicaţiei care se testează. Există două strategii de dezvoltare a cazurilor de test: o strategie bazată pe structura programelor şi o alta, bazată pe funcţionalitatea acestora.

Pentru testarea aplicaţiilor distribuite bazate pe Internet, trebuie luate în considerare următoarele aspecte:

- capacitatea de utilizare; uşurinţa în utilizare şi înţelegerea elementelor interfeţei constituie elemente importante în acceptarea aplicaţiei de către clienţi;
- funcţionalitatea; faptul că aplicaţia realizează ceea ce îşi propune prin specificaţii conduce la creşterea încrederii utilizatorilor în aceasta şi în firma producătoare;
- compatibilitatea; având în vedere faptul că există o multitudine de combinaţii între sisteme de operare, navigatoare şi aplicaţiile care rulează în navigatoare, asigurarea compatibilităţii aplicaţiei este o problemă importantă, deoarece există riscul ca o parte importantă dintre potenţialii utilizatori să nu poată utiliza aplicaţia datorită incompatibilităţilor şi astfel sunt pierduţi o serie de clienţi.
- performanţa; timpul de răspuns şi resursele ocupate reprezintă un aspect important pentru utilizatorii aplicaţiei.

În cazul aplicaţiilor Internet care accesează baze de date există o multitudine de cauze care duc la funcţionarea incorectă a acestora. De exemplu testul eşuat al unei tranzacţii într-o bază de date poate avea mai multe cauze:

- logica bazei de date; procedurile stocate, interogările sau comenzile de actualizare, inserare sau ştergere conţin erori;
- logica serverului de aplicaţii; există erori de proiectare sau de programare în cadrul funcţiilor implementate în serverul de aplicaţii;
- logica procesării tranzacţiilor; ordinea eronată a efectuării unor operaţii conduce la eşua-

rea întregii tranzacții;
- comunicația; în cazul unor probleme de comunicație la diverse niveluri, tranzacțiile fie nu au loc, fie se realizează incomplet sau incorect.

Testarea aplicațiilor pentru managementul portofoliilor proiectelor se realizează prin verificarea independentă a fiecărei componente, urmată de testarea întregii aplicații integrate. Pentru testare s-au stabilit ca obiective:
- testarea structurală a fiecărui modul astfel încât să se atingă o acoperire corespunzătoare pentru execuția instrucțiunilor
- testarea funcțională a aplicației realizată atât pentru fiecare modul în parte, cât şi pentru aplicația integrată. Este necesară stabilirea de cazuri de test pentru fiecare modul în parte;
- testarea conținutului astfel încât să fie respectate cerințele din specificații (de exemplu font cât mai mare, aşezarea simetrică în pagină, butoane de dimensiuni mai mari şi să fie vizibile, corectitudinea plasării informațiilor în formulare)
- testarea bazelor de date pentru fiecare componentă în parte, acolo unde este cazul;
- testarea securității tranzacțiilor în cazul transferului datelor senzitive, privind informații despre proiect, despre persoanele implicate, despre stadiul actual al derulării proiectelor pentru care se efectuează interogări;
- testarea performanțelor în diverse condiții de trafic, în special pentru transferul prin intermediul undelor radio.

Testarea structurală

Testarea structurală (*white box testing*) este o strategie de testare care necesită accesul la codul sursă şi la structura programului şi pune accentul pe acoperirea prin teste a căilor, ramificațiilor şi fluxurilor programului. Principalele metode de testare structurală au în vedere gradul în care cazurile de test acoperă sau execută codul sursă al programului. Principalele metode de testare structurală au în vedere gradul în care cazurile de test acoperă sau execută codul sursă al programului.

Strategiile de testare bazate pe căi utilizează fluxul de control al programului. Acestea reprezintă o familie de tehnici de testare bazate pe selectarea cu atenție a unei mulțimi de căi din program. Dacă mulțimea căilor este aleasă corespunzător, atunci se va atinge o anumită măsură a profunzimii testului. Pentru utilizarea acestor tehnici este necesară cunoaşterea completă a structurii programului şi accesul la codul sursă. Tehnicile sunt utilizate cel mai des de către programatori pentru testarea propriului cod. Cu ajutorul acestei tehnici se detectează erorile care cauzează execuția unei alte căi a programului decât aceea care trebuia să se execute.

Graful asociat programului este o reprezentare grafică a structurii de control al programului, care utilizează elemente ca proces, decizie şi joncțiune.

Tehnicile de testare bazate pe căile programului au la bază o serie de criterii de acoperire a codului care se testează precum: acoperirea instrucțiunilor, acoperirea ramificațiilor, acoperirea condițiilor, acoperirea ramificațiilor şi a condițiilor, acoperirea condițiilor multiple şi acoperirea tuturor căilor programului. Pe baza acestor criterii de acoperire a codului se determină seturile de date de test care se utilizează în testările structurale corespunzătoare: testarea instrucțiunilor, testarea ramificațiilor, testarea căilor etc.

Criteriul pentru acoperirea instrucțiunilor este ca fiecare instrucțiune să fie executată cel puțin o dată în cadrul unor teste. Dacă se realizează acest obiectiv, atunci declarațiile sunt acoperite în proporție de 100%. Acest lucru este echivalent cu o acoperire a nodurilor fluxului de control asociat programului în proporție de 100%. Criteriul de acoperire a instrucțiunilor este criteriul cel mai slab, deoarece cazurile de test identificate astfel încât să fie acoperite toate instrucțiunile iau în calcul doar acest criteriu, nefiind tratate toate situațiile posibile. Acest criteriu de acoperire se mai numeşte şi criteriul C1.

Acoperirea ramificațiilor are drept criteriu ca fiecare ramură a unei decizii să fie executată cel puțin odată. Se rulează un număr suficient de teste pentru a se executa toate ramificațiile din program cel puțin o dată. În cazul în care s-a realizat acest lucru, se atinge un procent de 100% a acoperirii ramificațiilor sau echivalent de acoperire a legăturilor dintre nodurile grafu-

lui asociat programului. Pentru software-ul structurat testarea tuturor ramificațiilor include şi acoperirea tuturor declarațiilor. Criteriul de acoperire a ramificațiilor se notează cu C2.
Acoperirea condițiilor are ca obiectiv identificarea de date de test astfel încât fiecare condiție cu rezultat logic să ia valorile posibile (adevărat sau fals).
Acoperirea tuturor căilor programului are ca obiectiv execuția tuturor căilor fluxului de control din program, care încep la intrarea în program şi se termină la ieşirea din program. Dacă se reuşeşte să se îndeplinească acest criteriu, atunci se acoperă căile în proporție de 100%. Acesta este cel mai puternic criteriu din familia de strategii de testare bazate pe căi şi este, în general, imposibil de atins deoarece numărul de căi poate ajunge foarte mare, mai ales prin existența structurilor repetitive.
Acoperirea declarațiilor şi a ramificațiilor sunt cerințe minime obligatorii pentru testarea structurală.
Pentru programele care conțin bucle, acoperirea marginilor interioare (boundary-interior cover) este deseori utilizată pentru a executa bucla cel puțin de două ori. În primul caz se execută bucla fără iterații, iar în al doilea caz se execută bucla cu una sau mai multe iterații. Dacă testarea se realizează utilizând doar analiza căilor sunt detectate circa 25% din erori.[MYER04]
Este necesară utilizării acestei strategii de testare în cazul aplicațiilor pentru MPP. Având în vedere complexitatea acestora, testarea structurală se realizează la nivel de modul (funcție, clasă), iar pentru integrare se vor urma alte strategii de testare.

Verificarea funcționalității aplicațiilor pentru MPP

Testarea funcțională (*black-box testing*) este o strategie de testare care necesită cunoaşterea comportamentului extern al programului pe baza specificațiilor. Testarea funcțională nu necesită cunoaşterea structurii interne a programului sau cunoştințe despre modul în care este implementat programul sau modulul.
Principalele tehnici de testare funcțională sunt testarea cu intrări aleatoare, partiționarea pe clase de echivalențe, analiza valorilor limită, graful cauză-efect şi ghicirea erorilor.
Testarea funcțională se desfăşoară pentru a constata dacă aplicația se comportă în conformitate cu specificațiile sale. Sunt avute în vedere:

- verificarea funcționalității ecranelor sau a paginilor;
- testarea formularelor;
- verificarea tranzacțiilor.

Testarea funcțională este efectuată atât pentru fiecare componentă în parte, cât şi pentru întreaga aplicație, prin integrarea componentelor:

- introducerea şi actualizarea datelor despre proiecte şi criteriile de selecție
- selecția proiectelor
- optimizarea portofoliului de proiecte
- situații şi rapoarte
- urmărirea proiectelor
- imprimarea
- culegerea de date şi calcularea de metrici.

Testarea funcțională se realizează comparând rezultatele obținute din rularea unei componente sau a aplicației cu rezultatele precizate prin specificații.
Prin *testarea conținutului* se urmăreşte corectitudinea şi aşezarea în cadrul ecranului a textelor, imaginilor şi controalelor din cadrul ferestrelor aplicației. Se remarcă următoarele posibile probleme:

- fontul utilizat pentru texte este necorespunzător;
- dimensiunea şi poziția controalelor (liste de selecție, butoane etc.) necorespunzătoare;
- necorespondența textelor cu cerințele specificate.

Testarea bazelor de date se realizează prin verificarea corectitudinii execuției interogărilor, a operațiilor de adăugare şi actualizare a datelor, precum şi verificarea conexiunilor dintre client, site-ul Web şi baza de date. Din testarea altor aplicații s-au constat erori datorate:

- interogărilor în care nu corespundea numărul de parametri;
- interogărilor în care tipurile datelor utilizate nu se potriveau;
- interogărilor în care apăreau nume de coloane ce nu existau în tabele;
- interogărilor care returnau rezultate incorecte;

- interogărilor care accesau toate datele şi nu pe cea căutată;
- închiderii conexiunii cu baza de date înaintea utilizării setului de date obţinut;
- existenţa unei anumite stări a setului de date.

Existenţa acestor rezultate, obţinute din testarea anterioară a unor aplicaţii, este utilă în ceea ce priveşte dezvoltarea şi testarea aplicaţiilor de MPP, acordându-se o atenţie sporită acestor aspecte generatoare de erori.

Testarea securităţii tranzacţiilor este necesar să se realizeze în special pentru partea de transfer a datelor personale ale utilizatorilor (nume, adresă, date de contact), a informaţiilor despre proiecte, despre resurse, despre stadiile actuale ale obiectivelor.

Testarea performanţele aplicaţiilor se realizează determinându-se comportamentul acestora în condiţiile unor viteze reduse de transfer a datelor, precum şi prin utilizarea de tehnologii de transmitere a datelor fără fir.

Concluzii

Testarea aplicaţiilor pentru managementul portofoliilor proiectelor necesită o atenţie deosebită, ţinând cont de complexitatea acestora. Pe lângă aspectele funcţionale ale acestor aplicaţii, se remarcă şi varietatea de tipuri de dezvoltarea a aplicaţiilor (de sine stătătoare, client-server, bazate pe Web şi aplicaţii mobile), precum şi existenţa unor platforme şi echipamente hardware diferite, fie pentru întreaga aplicaţie, fie pentru anumite componente ale acesteia.

Pentru fiecare tip de aplicaţie se utilizează metode şi tehnici specifice de testare, în lucrare fiind prezentate o parte din acestea.

În scopul realizării unei testări eficiente a aplicaţiilor pentru MPP, este necesară existenţa unui personal specializat şi instrumente pentru automatizarea acestui proces.

Bibliografie

[BEIZ90] Beizer, Boris – *Software Testing Techniques – Second Edition*, Van Nostrad Reinhold, New York, 1990

[IVAN99] Ivan, Ion, Pocatilu, Paul – *Testarea Software Orientat Obiect*, Editura Inforec, Bucureşti, 1999

[IVAN00] Ivan, Ion, Pocatilu, Paul, Sinioros, Panagiotis – *Testarea aplicaţiilor e-business*, Lucrările Simpozionului SIMPEC 2000, Vol. 2, Braşov, 2000

[MYER04] Glenford J. Myers - The Art of Software Testing, *Second Edition,* Revised and Updated by Tom Badgett and Todd M. Thomas with Corey Sandler, John Wiley & Sons, 2004

[PETE00] Peters, James F., Pedrycz, Witold – *Software Engineering – An Engineering Approach*, John Wiley & Sons, Inc, 2000

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

[POCA04a] Pocatilu, Paul – *Costurile testării software*, Editura ASE, Bucureşti, 2004

[POCA05] Paul POCATILU – *Metodologii de dezvoltare software,* în Informatica Economică, vol. IX, nr. 1(33), 2005, pp. 63-66

[POCA05a] Paul POCATILU – Project Portfolio Management Mobile Applications, în volumul The 7th International Conference on Informatics in Economy „Information & Knowledge Age", Bucureşti, 19-20 Mai 2005, pp. 1211-1216, ISBN 973-8360-04-8

[VLIE00] Vliet, Hans van, – *Software Engineering – Principles and Practice,* John Wiley & Sons, 2000