CALIBRAREA APLICAȚIILOR INFORMATICE

Ion Ivan
Cosmin Ivan
Mihai Marinescu

Catedra de Informatică Economică, A.S.E. - Bucureşti

Rezumat: Articolul prezintă concepte de bază pentru software, aplicații web și calibrarea bazelor de date. Se prezintă, de asemenea, modele de calibrare, bazate pe cost, precum și cerințele de calibrare. Subiectele următoarelor articole sunt: Calibrarea bazelor de date și Calibrarea aplicațiilor web.

Cuvinte cheie: calibrare, etalon, metrici, costuri, COCOMO, prototip, model

1. Calibrarea – concepte de bază

Termenul de calibrare este întâlnit ca atare în domenii foarte diferite:

- metrologie/măsurători: gradarea unui instrument în unități alese, în vederea efectuării de măsurători cu instrumentul respectiv;
- un pas important în analizele calitative cât și cantitative este calibrarea sau standardizarea; rezultatul metodei analitice și sensibilitatea echipamentelor de măsurare trebuie să fie calibrate sau standardizate;
- calibrarea termometrelor de mare precizie, bazate pe rezistor, se face folosind ca referință punctul de fierbere al oxigenului, precizia fiind de 0,0001 grade;
- ceasul atomic a avut un mare rol în dezvoltarea comunicațiilor în frecvențe de ordinul megahertzilor datorită posibilității de a calibra cu ajutorul lui cristalele oscilatoare;
- microsferele, particule de latex în diametru de 10 microni, produse de NASA în cadrul misiunilor spațiale, sunt folosite la calibrarea microscoapelor electronice;
- pentru calibrarea aparatelor de măsurare de mare precizie a impedanței, s-au dezvoltat 60 de standarde de impedanță;
- Biroul Național de Standarde din SUA deține un calculator digital de mare viteză (SEAC), folosit în numeroase domenii, precum și cel al calibrării termometrelor;
- prin calibrare, aparatele capătă noi performanțe: de exemplu, un galvanometru calibrat devine ampermetru;
- agricultură/agrotehnică: operație de sortare și de clasare a semințelor, fructelor, puieților etc. după mărime sau masă;
- tehnică: fază finală a unei operații de prelucrare mecanică (netezirea suprafeței etc.) cu ajutorul căreia se realizează cu mare precizie piese mecanice de aceleași dimensiuni;
- hidrotehnică: lucrare de restrângere a albiei unui râu;
- balistică: determinarea distanței de tragere (pt. tunuri, obuziere, mortiere etc.) prin măsurarea distanței până la locul de cădere a proiectilului.

Dintre definițiile de mai sus, se reține cea din metrologie, care se apropie cel mai mult de termenul de calibrare software, ce va fi dezvoltat în articolul de față.

Pentru a încadra calibrarea în ciclul de dezvoltare a produselor software, este necesară o prezentare succintă a acestora:

- **analiza** - ce se construiește;
- **proiectarea** - cum se construiește;
- **implementarea** - construirea propriu-zisă;
- **testarea** - cu asigurarea calității;
- **întreținere**.

Faza de analiză definește cerințele sistemului, independent de modul în care acestea vor fi îndeplinite. Aici, se definește problema pe care clientul dorește să o rezolve. Rezultatul acestei faze este documentul cerințelor, care trebuie să precizeze clar ce trebuie construit.

Faza de proiectare urmează fazei de analiză, pe baza cerinței căreia se stabilește arhitectura sistemului: componentele sistemului, interfețele și modul lor de comportare. Documentul de proiectare descrie planul de implementare a cerințelor. Se identifică detaliile privind limbajele de programare, mediile de dezvoltare, dimensiunea memoriei, platforma, algoritmii, structurile de date, definițiile de tip globale, interfețele etc.

Faza de implementare continuă construirea sistemului ori plecând de la zero, ori prin asamblarea unor componente pre-existente. Echipa trebuie să gestioneze problemele legate de calitate, performanță, biblioteci și debug. Scopul este producerea sistemului propriu-zis. O problemă importantă este îndepărtarea erorilor critice.

Faza de testare constă în verificarea aplicației atât ca întreg, cât și pe componente, și eventualele erori sau neconcordanțe cu cerințele clientului sunt înlăturate, corectate sau modificate. La testarea variantei finale, se va verifica de către client felul cum au fost puse în practică cerințele sale și felul cum sunt implementate funcționalitățile propuse în specificații

Faza de întreținere este cea mai îndelungată și se întinde pe toată durata de viață a produsului software, aplicația trebuind să fie întreținută la parametrii optimi, trebuie supravegheată și, în eventualitatea descoperirii unor alte erori decât cele descoperite în fazele de testare, acestea trebuind să fie înlăturate sau corectate. De asemenea, se obțin informații de la utilizatorul final privind modul de funcționare și posibilitățile de extindere a funcționalității și calității produsului software.

Încadrată între etapele clasice, de dezvoltare a aplicațiilor software, calibrarea se întrepătrunde cu fazele de analiză, proiectare și testare. Nu este o etapă de sine stătătoare, este prezentă sub forma unor procese desfășurate în cadrul etapelor amintite.

2. Calibrarea produselor software

Standardele ISO 9001, 9002 și 9003 se referă la sistemele de calitate după care se ghidează auditorii. Se aplică pentru multe tipuri de organizații de producție și manufactură, nu doar pentru software. Cel mai elaborat este ISO 9001, acesta fiind cel mai des folosit de organizațiile dezvoltatoare de software. Acoperă partea de documentație, design, dezvoltare, realizare, testare, instalare, service și alte procese. ISO 9000-3, diferit de ISO 9003, este un îndrumar pentru aplicarea standardului ISO 9001 de către organizațiile dezvoltatoare de software. Versiunea americană poate fi procurată direct de la ASQ (American Society for Quality) sau de la organizațiile ANSI. Pentru a fi certificat ISO 9001, un auditor evaluează organizația și oferă această certificare, de regulă, valabilă pe o perioadă de 3 ani, după care este necesară o reevaluare completă a organizației. Certificarea ISO 9000 nu implică obligatoriu calitatea produselor, indică doar că sunt implementate procese foarte documentate.

IEEE (Institute of Electrical and Electronics Engineers) creează standarde cum ar fi IEEE Standard for Software Test Documentation (Standard IEEE pentru documentația testării software) IEEE/ANSI Standard 829, IEEE Standard of Software Unit Testing (Standard IEEE pentru testarea modulelor software) IEEE/ANSI Standard 1008, IEEE Standard for Software Quality Assurance Plans (Standard IEEE pentru planuri ale asigurării calității produselor software) IEEE/ANSI Standard 730 și altele.

ANSI (American National Standards Institute – Institutul Național American de Standarde), principala instituție de standarde din Statele Unite; publică anumite standarde referitoare la software, conforme cu IEEE și ASQ (American Society for Quality – Societatea Americană pentru Calitate).

Standardul ISO 9000-3 descrie aplicarea unor caracteristici de calitate în procesul de dezvoltare a software-ului. În acest context, verificarea și validarea sunt definite astfel:

- verificarea este procesul de evaluare a produselor într-o anumită fază de dezvoltare, pentru a asigura corectitudinea și consistența acestora, relativ la standardele și produsele etalon existente pentru acea fază;
- validarea este procesul evaluării software pentru a asigura alinierea la cerințele impuse;

Verificarea și validarea sunt strâns legate de conceptul de calibrare software. Aproape în totalitate, metricile software ale calității se regăsesc ca subiecte ale calibrării. Aducerea unui produs la nivelul standardelor ISO presupune, pe de o parte, ca parametrii de calitate a etalonului să fie la nivel ISO și, pe de altă parte, coerența și corectitudinea metodelor de calibrare.

ISO 9000-3 mai descrie și conceptul de plan de calitate, ce conține:

- obiectivele de calitate;
- verificarea și validarea activităților;
- criteriile pentru intrările și ieșirile produsului în fiecare faze;
- planificarea detaliată a testării, verificării și validării activităților.

Calibrarea software presupune fie o serie teste manuale, fie o serie de scripturi sau cicluri de testare automată, relativă la medii de funcționare diferite, în urma cărora se realizează o serie de ajustări ale produsului software, în scopul atingerii unui nivel stabilit pentru anumite metrici (în general, metrici legate de calitate).

Aceste medii de funcţionare diferite pot fi sisteme de operare diferite (de exemplu, platforme diferite – Windows, AIX, SunOS/Solaris etc.), diferite versiuni de sisteme de operare (Windows 2000/NT/XP, Windows 9x/ME), diferite versiuni de surse de date ODBC (Oracle 8, 9i, 10g, SQL Server 6, SQL Server 2000), diferite versiuni de HTML (1.0, 2.0, 3.0, 4.0, extensii ale acestuia), diferite drivere de dispozitive şi aşa mai departe.

Totodată, esenţială în cadrul metodelor implicate în calibrare este compararea, prin care se verifică în paralel, în diverse medii descrise mai sus, comportarea atât a aplicaţiei care face obiectul calibrării, cât şi a aplicaţiei etalon, întocmindu-se tabele comparative cu diverşi parametri. În urma analizei acestor tabele, se stabileşte necesitatea implicării sau nu a altor metode de calibrare şi ajustare în procesul de dezvoltare.

3. Calibrare şi metode de cost

Pentru a aprecia costurile unui proiect software, un cuvânt foarte greu îl are experienţa celui care face această evaluare. Faptul de a fi văzut mai multe proiecte în desfăşurare şi de a fi evaluat a posteriori mai multe proiecte reprezintă un set de informaţii ce au o foarte mare valoare în procesul de evaluare. Cu toate acestea, evaluarea a posteriori trebuie privită cu grijă şi discernământ - un proiect realizat într-un timp mai scurt, dar nu sub o anumită limită minimă, va fi mai puţin finisat, mai puţin testat, mai sărac în funcţionalităţi de natură, în timp ce un proiect realizat într-un timp mai lung nu va avea aceste neajunsuri. Un proiect software complex nu este niciodată terminat. El poate suferi noi îmbunătăţiri, perfecţionări, creşteri de performanţă, adaptări care, într-un cuvânt, reprezintă procesul de calibrare din ciclul de viaţă al acelui proiect. De aceea, produsele software pot avea numeroase versiuni şi se află într-o continuă evoluţie.

Există, desigur, un anumit stadiu de la care produsul software devine utilizabil (momentul în care el răspunde cerinţelor specificate în caietul de sarcini) şi, deci, se poate considera proiectul încheiat. O dezvoltare, o îmbunătăţire, va fi un nou proiect. Când se face evaluarea a posteriori a unui proiect, în scopul de a-l calibra şi de a culege învăţăminte utile pentru evaluarea a priori a altor proiecte, va trebui să se ia în considerare numai acele costuri minimale, implicate de proiect, pentru a satisface cerinţele stipulate în caietul de sarcini.

Evaluarea trebuie să ţină cont şi de posibilităţile de plată ale clientului. Pornind de la un preţ impus (atunci când acesta se cunoaşte), se poate face evaluarea astfel încât proiectul să se încadreze în această limitare.

Strategiile generale de evaluare, în vederea calibrării, se pot clasifica în cele două grupe cunoscute: estimare de sus în jos (top down), în care se porneşte de la o estimare globală şi se rafinează prin detaliere, sau estimare de jos în sus (bottom-up) în care se porneşte de la costurile elementelor mărunte. De fapt, cele două metode se combină în felul următor:

- P1. se face o analiză top-down a sarcinilor implicate de proiect până la gradul de detaliere la care se poate ajunge;
- P2. se evaluează aceste sarcini „mici";
- P3. se construiesc costurile bottom-up;
- P4. se ajustează sumele totale redistribuind (de sus în jos) costurile pe sarcini.

Paşii P3 şi P4 se pot repeta de mai multe ori.

Tehnicile propriu-zise de evaluare sunt de două feluri: tehnici sau modele de evaluare prin descompunere şi tehnici sau modele algoritmice. Tehnicile de evaluare prin descompunere pun accentul pe descompunerea proiectului în task-uri cât mai mici şi evaluarea lor separată. Tehnicile de evaluare algoritmice pun accentul pe utilizarea unui anumit algoritm şi a anumitor formule pentru a stabili o legătură între cerinţe şi costuri. Şi tehnicile algoritmice pornesc tot de la nişte evaluări primare, bazate pe experienţă şi care se pot face tot prin descompuneri.

În evaluarea costurilor unui proiect software, o mare importanţă o are calibrarea metodei utilizate. Prin calibrare se înţelege utilizarea metodei în condiţii asemănătoare pe un număr de proiecte (cel puţin 2-3). Prin condiţii asemănătoare se înţelege: aceeaşi companie elaboratoare, aceleaşi echipe (sau aceiaşi specialişti), aceleaşi instrumente de dezvoltare. Prin aplicarea unor metode necalibrate, rata erorilor poate fi de aproximativ 3-4 ori mai mare, decât prin folosirea metodelor orientate pe calibrare. Nici o metodă (calibrată sau nu) nu poate garanta o eroare zero.

Pentru a putea măsura proiectele software, se folosesc mai multe metrici. O metrică pentru un proiect software este o metodă de a caracteriza, din punct de vedere cantitativ, un proiect software.

Modelul Effort Estimation (Estimarea efortului) de evaluare a proiectelor software este un model prin descompunere. El constă în descompunerea proiectului pe două coordonate: se descompune mai întâi proiectul în task-uri funcţionale mici, apoi fiecare task funcţional este descompus, la rândul său, în subtask-urile

corespunzătoare fazelor globale ale unui proiect. Task-urile funcționale sunt caracteristice proiectului. Task-urile corespunzătoare fazelor globale ale proiectului sunt: analiza, concepția preliminară, concepția detaliată, codificarea, testarea. Se estimează costurile pentru fiecare subtask. Se vor putea apoi determina costurile pe funcții sau pe faze și costul total.

În această reprezentare, se pot evidenția mai clar și costurile pe faze, lucru important deoarece, deseori, costurile unitare sunt diferite pe tipuri de faze diferite. De exemplu, de obicei, cost analist > cost conceptor > cost codificator > cost testor. Toate aceste costuri se exprimă în monedă/om zi.

Această metodă de evaluare este relativ simplă și foarte eficientă, ea ducând la rezultate foarte bune, cu grad de eroare foarte mic. Pentru a fi aplicată este, însă, necesară o anumită experiență.

Modelele Lines of Code (linii de cod) și Function Point (puncte funcționale) sunt, de asemenea, modele de evaluare prin descompunere.

Se prezintă modelul LOC. Ca și modelul EE de evaluare, se pornește de la descompunerea proiectului în task-uri funcționale. Se evaluează apoi volumul fiecărei funcții în LOC. Cunoscându-se productivitatea medie în LOC/zi, se determină durata de elaborare pentru fiecare funcție. Cunoscându-se costul unitar în cost/LOC, se determină costul pentru fiecare funcție. Prin însumare, se pot apoi determina durata totală și, respectiv, Costul total. Durata efectivă va fi, desigur, mai mică dacă vor lucra mai mulți oameni în paralel.

Aplicarea modelului LOC trebuie să țină cont și de faptul că productivitatea depinde de limbajul de programare folosit (se scriu mai multe linii de cod assembler, decât linii de cod C++, de exemplu). Modelul LOC se aplică mai dificil când este vorba de utilizarea unor scule vizuale de elaborare. În acest caz, modelul FP este mai potrivit.

Modelul COCOMO (COnstructing COst MOdel) „de bază" a fost dezvoltat de B. W. Boehm.

Se prezintă pașii ce trebuie urmați în aplicarea acestui model.

- P1. Se încadrează proiectul de realizat într-una dintre următoarele trei tipuri de program: organic, embeded și semidetached. Pentru a realiza acest lucru se pornește de la următoarele patru caracteristici: dimensiune, noutate, constrângeri de timp, mediu de lucru. În general, un proiect organic este de complexitate mică, un proiect embeded este de complexitate mare, iar un proiect semidetached este de complexitate medie;
- P2. Conform tipului de proiect stabilit, se aleg constantele (notate a, b, c, d) de calcul ce vor fi folosite ulterior;
- P3. Se aplică formulele (care sunt numerice, și nu dimensionale):

$$E = a * KLOC_b$$

$$D_d = c * E_d$$

$$N = E_b / D_d$$

unde:

E – efort pentru realizarea proiectului exprimat ca număr persoane lună;

KLOC – mii linii sursă cod;

D – durată exprimată în luni;

N – numărul de persoane care lucrează la proiect.

O perfecționare a modulului COCOMO de bază este **modelul COCOMO intermediar**.

În scopul de a aduce o caracterizare mai fină a proiectului, modelul COCOMO intermediar adaugă un număr de factori de ajustare a efortului necesar pentru realizarea proiectului. Acești factori sunt repartizați în 4 grupe: atribute produs, atribute hardware, atribute ale persoanelor care realizează proiectul, atribute ale proiectului însuși. Fiecare factor are maximum 6 categorii posibile de valori notate simbolic cu VL (foarte jos), LO (jos), NM (normal), HI (înalt), VH (foarte înalt), XH (extrem de înalt). Fiecare categorie are o anumită valoare efectivă, pentru un anumit factor de ajustare a efortului.

Modelul COCOMO detaliat este asemănător cu modelul COCOMO intermediar. Diferența constă în folosirea de factori de ajustare a efortului, diferiți pe fiecare fază a unui proiect. Sunt definite 6 faze: cerințe, proiectare produs, proiectare detaliată, codificare și teste unitare, teste de integrare, întreținere.

Modelul REVIC face parte, de fapt, tot din familia COCOMO. El a fost dezvoltat de Raymond L. Kyle de la Hughes Aerospace. Modelul adaugă încă un alt tip de proiect (proiecte ADA, specifice domeniului aviației, care

se foloseşte mai mult de limbajul ADA) precum şi alte elemente pentru calculul fae: Required Reusability (măsoară efortul suplimentar, necesar pentru a generaliza modulele software în aşa fel încât să se poată reutiliza în alte aplicaţii), Requirements Volatility (Măsoară volumul de muncă necesar pentru a reface proiectarea produsului ca urmare a unei eventuale modificări în specificaţiile clientului - implică măsura timpului folosit pentru evaluarea cerinţelor de modificare, estimare a impactului asupra procesului de elaborare, modificare a condiţiilor contractuale), Risk (permite adăugarea unui coeficient care ia în considerare diversele niveluri ale riscului), Classified Security Application (măsuri ce trebuie luate în desfăşurarea activităţii, pentru a răspunde unor cerinţe de securitate deosebite, în sensul de secretizare).

Modelul Putnam este derivat din studierea unor proiecte mari. Foloseşte curbele Raileigh-Norden (descrise analitic de Raighley şi completate cu date empirice culese de Norden). Acest model porneşte de la ideea că există o anumită distribuţie a efortului, pe parcursul unui proiect, în raport cu diversele sale faze. Efortul de dezvoltare K (în persoane-an) este, conform acestui model, dat de formula:

$$K = (dim / (C * T^{4/3})) * 3$$

unde:

- dim – dimensiunea proiectului în KLOC;
- C – constantă tehnologică (combină efectul instrumentelor de dezvoltare, limbajelor, metodologiilor, asigurării calităţii etc.); variază între 2000-11000 (de exemplu, C = 2000 pentru mediu de dezvoltare sărac, C = 8000 pentru mediu de dezvoltare potrivit şi C =11000 pentru mediu de dezvoltare excelent);
- T - timpul de dezvoltare în ani.

Până acum, au fost prezentate elemente privind metricile şi evaluarea costurilor pentru proiectele software. Aceste elemente pot fi folosite ca atare sau prin intermediul unor aplicaţii specializate.

Există numeroase astfel de aplicaţii atât comerciale, cât şi din domeniul public, precum: ACE_IT realizat de Armata şi Marina Americană (www.aceit.com), COSTAR realizat de SoftStar Systems (www.softstarsystems.com), Cost*Xpert realizat de Marotz, Inc. (www.marotz.com), Price_S realizat de Galborath (www.gaseer.com), Knowledge Plan şi Check Point realizate de Software Productivity research (www.spr.com), Estimate Profesional realizat de Software Productivity Center (www.spc.ca/estimate/ index.htm), GA SEER realizat de Galborath (www.gaseer.com), SLIM - Software Lifecycle Management (Managementul Ciclului de Viaţă al Produselor Software) realizat de Quantitative Software Management (Management Catitativ al Produselor Software) (www.qsm.com), Price_S realizat de Price Systems (www.pricesystems.com).

Aplicaţiile din domeniul public provin, în general,de la universităţi cum ar fi:

- COCOMO 2 - de la University of Southern California (sunset.usc.edu/COCOMOII/cocomo.html),
- COSMOS (www-cs.etsu.edu/faculty/ henryj/ dsstud/cosmos.exe),
- SEAT (www-cs.etsu.edu/faculty/henryj/ dsstud/seat/seat24_2.zip), de la East Tenessee State University.

Activitatea de evaluare a unui proiect precede realizarea propriu-zisă a proiectului. Diversele unelte de evaluare este bine să permită ca informaţiile legate de evaluare să fie folosite şi în timpul desfăşurării proiectului. În felul acesta, se poate urmări mai uşor în ce măsură proiectul se încadrează în costurile estimate, unificându-se evaluarea cu planificarea şi urmărirea.

4. Calibrarea modelului

În cadrul calibrării aplicaţiilor informatice, un rol important o are calibrarea modelului, a prototipului aplicaţiei. Ca dezvoltator de produse software de interes general, o strategie în crearea produselor software o constituie dezvoltarea unui produs cu complexitate maximă, ce rezolvă un număr maxim de probleme dintr-un anumit domeniu. Produsul este structurat în module pentru o mai bună gestionare.

Ieşirile sunt generate de modulele produsului, fiecare modul generând o anumită grupă de ieşiri. Generarea acestor rezultate necesită anumite intrări din partea utilizatorului.

Deoarece marea majoritate a posibililor clienţi nu au nevoie de această complexitate ridicată a ieşirilor, dar fiecare are nevoie să rezolve anumite probleme bine determinate, este necesară calibrarea prototipului produsului software pe cerinţele fiecăruia.

Calibrarea este văzută ca o negociere între producătorul de software şi viitorul utilizator, cu scopul de a stabili performanţele produsului, în raport cu un nivel optim al costului calităţii.

În această negociere, se porneşte de la ieşirile dorite de client. Dezvoltatorul produsului software îi prezintă clientului modulele pe care aceasta ar trebui să le achiziţioneze în vederea obţinerii ieşirilor dorite şi de ce intrări va avea nevoie aplicaţia.

Intrările acestea presupun costuri de obţinere şi de introducere din partea utilizatorului şi există situaţii când un utilizator nu este dispus să le furnizeze pe toate. Plecând de la intrările pe care acesta le poate furniza, din multitudinea intrărilor necesare, se stabilesc modulele ce pot fi utilizate şi achiziţionate de client şi ieşirile ce vor fi realizate de produsul software.

Aceste module necesită un efort financiar din partea clientului la achiziţionare, efort pe care acesta s-ar putea să nu îl poată susţine.

De asemenea, pot exista date de ieşire, generate de combinaţia de module, şi intrări selectate, pe care utilizatorul nu le-a cerut iniţial. Este util ca dezvoltatorul produsului informatic să îi prezinte clientului şi aceste ieşiri ce pot fi utile în viitor.

Cu toate transformările aduse de calibrare modelului iniţial, este necesar să se ţină cont de păstrarea scalabilităţii acestuia. Calibrarea unui astfel de prototip al unui produs software înseamnă, de fapt, obţinerea unui maxim de rezultate prin minim de informaţii de intrare.

5. Calibrarea aplicaţiilor web

Calibrarea aplicaţiilor bazate pe arhitectura Web, în plus faţă de calibrarea aplicaţiilor clasice, necesită o serie de teste specifice cum ar fi: testarea de încărcare, testarea de compatibilitate, testarea serverului Web, testarea serverului de aplicaţii şi testarea bazelor de date.

Testarea de încărcare se utilizează pentru a verifica dacă site-ul Web poate gestiona un anumit număr de utilizatori care îl accesează concurent, în limite acceptabile ca timp de răspuns.

Prin testarea de compatibilitate, se urmăreşte aspectul şi comportamentul site-ului Web în raport cu o varietate de sisteme de operare şi de navigatoare pe Internet. Această testare scoate în evidenţă problemele cu controalele ActiveX, applet-urile Java, funcţiile JavaScript sau VBScript şi formulare din pagini. La ora actuală, există peste 100 de combinaţii posibile între diverse sisteme de operare Windows şi diverse versiuni ale navigatoarelor Netscape şi Internet Explorer.

Testarea serverului Web are în vedere testarea interacţiunilor dintre serverul Web şi serverul de aplicaţii, verificarea integrităţii bazei de date, în cadrul serverului de baze de date, verificarea faptului ca scripturile ASP, PHP sau JSP, aplicaţiile ASP.NET să se execute corect pe server.

Testarea serverului de aplicaţii se realizează ţinându-se seama de caracteristicile funcţionale şi structurale ale acestuia. Se testează componentele serverului, folosind metode clasice de testare, precum şi metode de testare ce iau în considerare tranzacţiile şi comunicaţiile asincrone dintre aceste componente.

Testarea bazelor de date presupune verificarea conexiunilor dintre site-ul Web şi baza de date. Prin testarea performanţelor, se măsoară comportamentul site-ului Web, în diverse condiţii de trafic.

În ceea ce priveşte aplicaţiile distribuite, bazate pe Web, există şi aici o mulţime de instrumente pentru testarea automată. Astfel de aplicaţii sunt eValid, Rational SiteCheck, SilkPerformer, LoadRunner şi, prin caracteristicile de a analiza încărcarea şi capacitatea serverului Web şi de a oferi o serie de indicaţii pentru reglaje fine ale site-ului, sunt instrumentele potrivite pentru a efectua şi calibrarea.

Calibrarea aplicaţiilor de comerţ electronic se realizează fie de către echipe specializate în testare, din cadrul departamentului de asigurare a calităţii al firmei, fie de către o firmă specializată (outsourcing). Elementele care stau la baza deciziei de contractare a unei firme specializate pentru a realiza calibrarea sunt dorinţa de asigurare a unei obiectivităţi asupra evaluării calităţii şi, nu în ultimul rând, analiza cost-beneficiu, realizată pe baza estimării costurilor privind calibrarea.

6. Calibrarea bazelor de date

Proiectarea bazei de date este o parte importantă a etapei de proiectare a aplicaţiei. În cadrul acestei, etape se elaborează modelul bazei de date, în conformitate cu cerinţele şi specificaţiile existente, se sistematizează tabelele (câmpuri, tipul acestora), legăturile dintre ele şi, eventual, integrarea în acest model a unor componente specifice SGBD (view-uri, proceduri stocate, trigger-e).

Totuşi, pentru acelaşi set de cerinţe şi de specificaţii (de obicei, relativ flexibile) pot fi elaborate diverse modele ale bazei de date, diferite între ele prin parametrii importanţi, cum ar fi gradul de normalizare (şi, implicit, numărul tabelelor şi conţinutul acestora), gradul de implementare al redundanţei etc. Aceste modele diferite între ele sunt corecte, însă numai unul sau câteva pot fi potrivite standardelor de calitate, impuse de dezvoltatori sau de beneficiari.

Totodată, aproape întotdeauna, în cadrul dezvoltării proiectelor software, diverse cerinţe şi specificaţii, inexistente în faza iniţială, apar „din mers". Integrarea neorganizată a acestora în cadrul proiectului determină aproape întotdeauna apariţia unor disfuncţionalităţi, generând costuri suplimentare atât dezvoltatorilor, cât şi beneficiarilor. În mod particular, componentele de lucru cu baze de date, din cadrul unei aplicaţii, şi, respectiv, bazele de date în sine sunt deosebit de „sensibile" în ceea ce priveşte integrarea unor modificări generate de cerinţe/specificaţii apărute pe parcursul dezvoltării ulterioare momentului de proiectare a modelului bazei de date.

Calibrarea bazelor de date rezolvă parţial atât problema alegerii modelului cel mai potrivit nivelului de calitate propus, cât şi problema integrării „din mers" a diverselor modificări în cadrul BD/ interfeţei cu BD.

Procesul presupune a priori existenţa unei aplicaţii ale cărei cerinţe şi specificaţii sunt puţin diferite de ale aplicaţiei în curs de dezvoltare şi ale cărei metrici software, legate de calitate (grad de fiabilitate, mentenabilitate, etc.) ating sau depăşesc parametrii propuşi pentru aplicaţia curentă. Modelul BD (cu parametrii săi) respectiv interfaţa BD a acestei aplicaţii vor servi ca model, ca *etalon* pentru dezvoltarea componentelor respective din cadrul primei aplicaţii.

În cazul (destul de curent întâlnit în dezvoltarea pe scară largă) în care se planifică dezvoltarea mai multor aplicaţii cu specificaţii sensibil identice, aplicaţia *etalon* va fi transformată în *prototip*, prin intermediul procesului de prototipizare care presupune:

1. eliminarea elementelor specifice aplicaţiei respective, în condiţiile în care se menţine funcţionalitatea ansamblului;
2. integrarea unor caracteristici posibil comune pentru mai multe aplicaţii planificate, îndeplinind cerinţa menţionată mai sus

7. Concluzii

Calibrarea produselor informatice este un proces complex, ce are o importanţă deosebită în dezvoltarea de produse software. Efectele calibrării se văd imediat şi sunt şi de durată (durata de viaţă a aplicaţiei ce s-a dezvoltat).

Implementarea procesului de calibrare conduce la crearea unor aplicaţii ce au următoarele caracteristici:

- răspund foarte bine cerinţelor clienţilor, ţinând cont de un nivel optim al costului calităţii;
- se încadrează în costurile estimate iniţial.

Calibrarea permite reproducerea sau chiar depăşirea parametrilor de calitate ai unui produs similar, de succes, dezvoltat anterior, şi asigură că nivelurile de calitate şi de cost propuse pot fi planificate şi controlate cu exactitate

În concluzie, calibrarea este esenţială pentru producţia pe scară largă de aplicaţii informatice, atunci când se are în vedere ca produsul rezultat să aibă un nivel de calitate cât mai ridicat, obţinut la un nivel al costurilor totale cât mai scăzut.

Bibliografie

1. **DIACONESCU, ŞT.**: Evaluarea costurilor proiectelor software. În: PC Report Nr. 82, Iulie, 1999.
2. **EBRAHIMI, N. B.**: How to Improve the Calibration of Cost Models. În: IEEE Transactions on Software Engineering, January – February 1999, Vol. Nr. 25, Nr. 1.
3. **FEHLMANN, T.M.**: Combinatory metrics for software development, QFD Institut Deutschland, 2002.
4. **IVAN, I., P. POCATILU**: Testarea Software Orientat Obiect, Editura Inforec, Bucureşti, 1999.
5. **ZUSE, H.**: A framework of Software Measurment, Verlages Walter de Gruyter, Berlin, 1998.