34. INSPECŢIA SOFTWARE

34.1 Auditul informatic

În prezent, utilizarea calculatorului şi a programelor informatice a devenit un element vital în cadrul sistemului informaţional al întreprinderilor. Odată cu integrarea noilor tehnologii informaţionale în procesele de prelucrare, transmitere şi stocare a datelor, au apărut o serie de ameninţări şi vulnerabilităţi ale sistemului informaţional. Astfel, managerii şi-au pus problema garantării corectitudinii şi securităţii operaţiilor din sistemul informatic şi convergenţei acestora cu obiectivele şi strategiile organizaţiei.

Pentru a contracara aceste aspecte negative s-a impus elaborarea unor proceduri de control a sistemelor informaţionale la nivelul fiecărei întreprinderi. Aceste obiective şi proceduri de control intern au ca scop asigurarea securităţii sistemelor informaţionale şi reducerea riscului pe care l-ar putea avea orice ameninţare şi vulnerabilitate asupra sistemului.

În literatura şi practica din România se întâlnesc termenii de *auditul sistemelor informaţionale*, *auditul sistemelor informatice* sau *auditul informatic*, *auditul IT*. Diferenţa conceptuală dintre acesti termeni este dată pe de o parte de conţinutul şi nivelul la care se desfăşoară activitatea de audit şi pe de altă parte de diferenţa conceptuală dintre noţiunile de sistem informaţional şi sistem informatic. Astfel, auditul sistemului informaţional este, conceptual, cel mai cuprinzător, acoperind prin obiectivele sale toate nivelurile sistemului informaţional, de la evaluarea proiectării şi utilizării sistemului informatic, până la evaluarea politicilor şi procedurilor de securitate de la nivelul operaţional şi strategic. Auditul sistemului informatic, respectiv auditul informatic, acoperă prin obiectivele sale doar sistemul informatic.

Auditul informatic se referă la evaluarea riscurilor informatice ale securităţii fizice, securitatea logică, managementul schimbărilor, planul de asistenţă etc. În cazul general auditul informatic reprezintă un ansamblu de procese informatice pentru a răspunde la o cerere precisă a clientului.

Principalele tipuri de audit informatic sunt [Ivan05]:

- *auditul sistemului operaţional de calcul* – se referă la revizia controalelor sistemelor operaţionale de calcul şi reţelelor, la diferite niveluri: sistem de operare, reţea, software de aplicaţie, baze de date;
- *auditul instalaţiilor IT* – este legat de securitatea fizică, controalele mediului de lucru, sistemele de management şi echipamentele IT;
- *auditul sistemelor aflate în dezvoltare*, care se ocupă cu controalele managementului proiectului, specificaţiile, dezvoltarea, testarea, implementarea şi operarea controalelor tehnice şi procedurale;
- *auditul managementului IT* – include revizia organizaţiei, structurii, strategiei, planificării muncii, planificării resurselor, stabilirii bugetului, controlul costurilor etc;

- *auditul procesului IT* – cuprinde revederea proceselor care au loc în cadrul IT cum sunt dezvoltarea aplicaţiei, testarea, implementarea, operaţiile, mentenanţa, gestionarea incidentelor;
- *auditul managementului schimbărilor* – se ocupă cu verificarea planificării şi controlului schimbărilor la sisteme, reţele, aplicaţii, procese etc., cuprinde managementul configuraţiei, controlul codului de la dezvoltare, prin testare, la producţie şi managementul schimbărilor;
- *auditul controlului şi securităţii informaţiilor* – se referă la verificarea controalelor referitoare la confidenţialitatea, integritatea şi disponibilitatea sistemelor şi datelor;
- *auditul conformităţii cu legalitatea* – include copyright, conformitate cu legislaţia, protecţia datelor personale;
- *auditul accidentelor dezastruoase/planificării continuităţii afacerii/refacerii după dezastre* – cuprinde reviziile măsurilor propuse pentru restaurarea după un dezastru care afectează sistemul şi verificarea modului în care organizaţia abordează managementul riscurilor;
- *auditul strategiei IT* - include revizia aspectelor variate ale strategiei IT, viziune şi planuri, precum şi relaţiile cu alte strategii, viziuni şi planuri.

Un loc important în cadrul auditului informatic îl ocupă *inspecţia software*.

34.2 Necesitatea şi obiectivele inspecţiei software

Calitatea constituie un ţel principal în industria software. Încă se manifestă părerea că nu există un mijloc eficient pentru îmbunătăţirea calităţii fără lungirea ciclului de dezvoltare şi/sau creşterea costurilor. Încadrarea asigurării calităţii software într-un program şi un buget prestabilite este aproape imposibil de realizat. Scurtarea ciclului de dezvoltare, limitarea resurselor şi creşterea complexităţii software duc la o scădere a calităţii sale şi la creşterea numărului de defecte. Impactul economic al defectelor este deosebit. Ele reprezintă principala cauză a eşuării aplicaţiilor şi provoacă pagube însemnate atât utilizatorilor cât şi organizaţiei dezvoltatoare.

Este posibil ca anumite defecte şi vulnerabilităţi să treacă neobservate, inclusiv în etapa de testare, mai ales când consecinţele lor nu sunt atât de vizibile. Ele pot rămâne nedetectate până în etapele de implementare şi de exploatare când, pentru remedierea lor, sunt necesare resurse suplimentare foarte mari.

Un mijloc important pentru creşterea calităţii unui produs software îl constituie îmbunătăţirea testării sale. Testarea este recunoscută ca un punct critic în asigurarea calităţii totale, dar are limitele ei, precum:

- crearea, execuţia, validarea şi întreţinerea seturilor de date şi a procedurilor de testare este scumpă şi consumatoare de timp;
- gradul de acoperire – procentul de instrucţiuni testate – scade considerabil pe masura creşterii complexităţii produsului;
- adesea poate fi dificilă şi de durată determinarea cauzei reale care provoacă o eroare, astfel încât dezvoltatorii să localizeze exact secvenţa de cod ce trebuie modificată;

- testarea nu poate acoperi toate bug-urile posibile – studii realizate în acest sens arată că, de obicei, testarea duce la înlăturarea a mai puţin de 50% din defecte; chiar şi cele mai bune procese de testare nu înlătură mai mult de 85% din defecte [Reas**].

În concluzie, deşi testarea este o etapă importantă în asigurarea calităţii unui produs software, nu reprezintă un panaceu. Testarea singură nu poate garanta eliminarea tuturor defectelor şi nici nu asigură un nivel suficient de înalt al calităţii.

"*Testarea nu se termină niciodată, este doar abandonată*" [Onei01a]. Pentru a înţelege acest citat trebuie făcută distincţia între conceptul de cod şi cel de acoperire. Chiar dacă este asigurată întreaga acoperire a codului, acest lucru nu garantează înlăturarea tuturor defectelor. Testarea completă cere acoperirea tuturor ramurilor, nu numai a instrucţiunilor ca atare. Acoperirea tuturor ramurilor este mai dificil de realizat decât acoperirea totală a codului. Codul unui sistem complex poate conţine milioane de ramuri. Chiar dacă ar fi posibilă construirea unui sistem complet de testare, timpul şi costul necesar execuţiei ar fi prohibite.

Deoarece testarea nu asigură acoperirea totală şi necesită dezvoltarea de instrumente scumpe, sunt necesare alte tehnici pentru asigurarea unei înalte calităţi a produselor software. Una dintre aceste tehnici este inspecţia software, care acoperă multe din limitele testării.

Inspecţia software este un proces de revizuire tehnică realizat de-a lungul etapelor de dezvoltare a produselor, cu scopul de a identifica şi elimina defectele. Ea se poate aplica oricărui produs rezultat în diversele etape ale procesului de dezvoltare: determinarea cerinţelor, proiectare, codificare, chiar şi în etapa de testare.

Comunitatea dezvoltatorilor de software ştie de mult că inspecţia software constituie o tehnică eficientă pentru înlăturarea defectelor, cu beneficii substanţiale pe termen lung. Inspecţia software este benefică deoarece identifică şi înlătură erorile critice încă din primele etape ale ciclului de dezvoltare, înainte de a se ajunge în etapa de testare sau implementare.

Inspecţia software – procesul de examinare a fiecărei linii a codului sursă pentru identificarea defectelor şi vulnerabilităţilor – este o practică obişnuită în multe organizaţii. Ea are ca scop detectarea şi corectarea defectelor şi prevenirea propagării lor de-a lungul ciclului de dezvoltare.

Specialiştii cu experienţă ştiu că dezvoltarea produselor software este un proces de experimentare care implică descoperirea continuă de informaţii tehnice asociate cu funcţionalitatea, forma şi corectitudinea produsului. Inspecţia software reprezintă o practică integrată în acest proces de experimentare. Ea eficientizează etapele de dezvoltare şi testare prin reducerea efectelor cauzate de defecte şi vulnerabilităţi, în paralel cu creşterea nivelurilor de securitate, integritate, încredere, disponibilitate, viabilitate şi mentenanţă ale produsului.

Inspecţia software este o tehnică de detectare a defectelor din produsele software înainte ca acestea să fie implementate. Ea a fost introdusă la compania IBM de către Michael Fagan în 1976 [Faga76], cu scopul de a îmbunătăţii calitatea produselor software şi de a creşte productivitatea. De atunci s-a dezvoltat continuu şi se referă la un proces structurat prin care se încearcă identificarea defectelor în documentele întocmite în diversele etape ale procesului de dezvoltare. Procesul de

dezvoltare a produselor software constă dintr-o serie de etape care se finalizează cu un anumit produs: definirea cerinţelor, proiectare, codificare, testare şi mentenanţă. Identificarea defectelor este necesar să se facă cât mai devreme posibil, deoarece costurile de remediere a unui defect sunt de 10 până la 100 ori mai mici în primele etape faţă de remedierea lor în etapa de mentenanţă. Acest lucru se realizează prin inspectarea ieşirilor din fiecare etapă în parte şi compararea lor cu cerinţele corespunzătoare, sau prin îndeplinirea unui criteriu de terminare a etapei respective.

Inspecţia este cea mai eficientă metodă de evaluare software şi se deosebeşte de alte forme de evaluare prin aceea că este un proces riguros, bine definit pentru examinarea în detaliu a produselor software şi identificarea defectelor lor. Numită şi inspecţie Fagan sau inspecţie formală, ea defineşte un proces ca fiind o activitate specifică cu criterii de intrare şi de ieşire predefinite. Criteriile de intrare sunt cerinţele ce trebuie îndeplinite pentru a se începe un anumit proces, iar criteriile de ieşire sunt cerinţele ce trebuie îndeplinite pentru terminarea procesului. Pentru fiecare activitate, inspecţia software stabileşte dacă ieşirea din proces este conformă cu criteriile de ieşire aferente procesului [Faga86]. Orice abatere de la aceste criterii este considerată un defect.

Inspecţia software sau evaluarea codului este o examinare vizuală a codului sursă cu scopul identificării defectelor şi/sau nerespectării standardelor de codificare. Ea urmăreşte garantarea completitudinii şi corectitudinii codului sursă şi asigurarea faptului că implementarea codului este în concordanţă cu specificaţiile software. În acest fel se asigură eliminarea erorilor încă din fazele incipiente ale ciclului de dezvoltare şi creşterea calităţii produselor software.

Este important de subliniat faptul că inspecţia software nu este acelaşi lucru cu testarea, deşi ambele urmăresc asigurarea unei înalte calităţi a produsului, între ele existând anumite diferenţe, dintre care se pot aminti:

- când se testează, se execută cod; când se inspectează, se evaluează cod;
- testarea se face în etapa de testare; inspecţia software se face în etapa de codificare, dar şi în etapele anterioare (determinarea cerinţelor, proiectare);
- prin testare nu se parcurg toate ramurile programului; prin inspecţie software se descoperă defecte pe ramuri executate foarte rar şi puţin probabil de inclus în strategia de testare;
- defectele identificate în etapa de inspecţie software sunt înlăturate imediat; testarea se caracterizează printr-o abordare serială: întâi se observă efectele, apoi se caută cauza pentru înlăturarea ei;
- inspecţia software nu execută cod, deci este independentă de hardware, nu cere resurse sistem sau modificări în comportamentul programului şi poate fi aplicată cu mult timp înainte ca resursele hardware să fie disponibile pentru testare.

În figura 34.1 se prezintă încadrarea inspecţiei în ciclul de dezvoltare a produsului software.

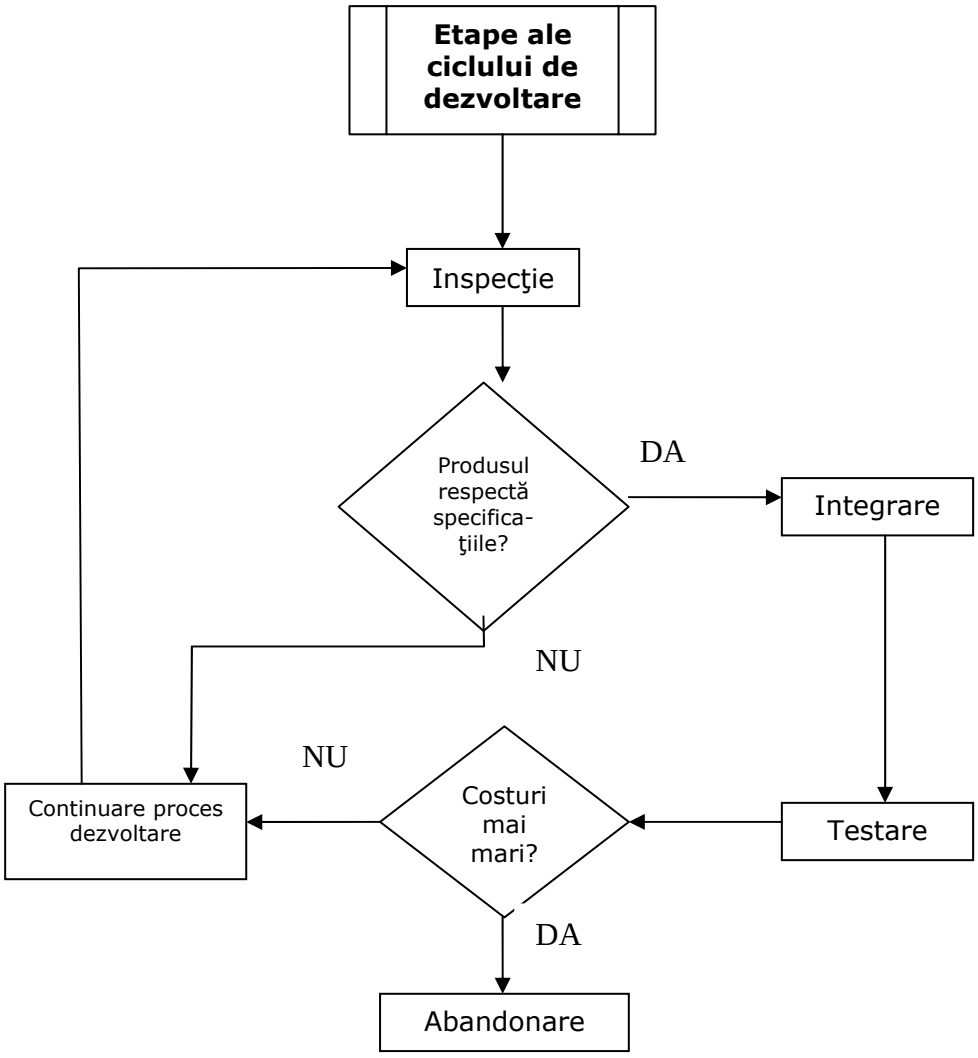


Figura 34.1 *Locul inspecţiei în ciclul de dezvoltare a software.*

Inspecţia software implică interacţiunea următoarelor elemente:

- etapele inspecţiei software;
- rolurile participanţilor;
- colecţia de procese şi date;
- produsul inspectat;
- infrastructura corespunzătoare.

Ea ajută organizaţiile producătoare de software să realizeze produse mai bune. Prin aplicarea ei în fiecare etapă a ciclului de dezvoltare, se asigură o bază tehnică corectă pentru etapele următoare. Descoperirea şi corectarea timpurie a defectelor reduce resursele necesare pentru depanare în etapele ulterioare ale proiectului, ducând la reducerea costurilor totale de realizare a produsului. Rezultatele obţinute la compania IBM [Faga86] arată că pot fi detectate 80-90% din defecte, obţinându-se o reducere cu până la 25% a costurilor. În acelaşi timp se asigură o eficienţă sporită testării, timpul necesar acestei activităţi reducându-se substanţial. Un alt avantaj îl reprezintă evaluarea imediată şi feedback-ul primit de autor după fiecare etapă a procesului de dezvoltare.

Calitatea codului care a făcut subiectul inspecţiei software este de nivel înalt, ceea ce duce la reducerea timpului şi efortului de testare.

Deoarece inspecţia software este o examinare a codului, în această etapă nu este necesară execuţia şi testarea sa.

Dupa definiţia dată de Fagan [Faga86], "*un defect este o situaţie în care o anumită cerinţă nu este îndeplinită*".

Defectele identificate în procesul de inspecţie software se clasifică în două categorii: defecte majore şi defecte minore. Defectele care constau în funcţionalităţi sau specificaţii incorecte sau lipsă se încadrează în categoria defectelor majore. Dacă nu sunt înlăturate, produsul nu va funcţiona corect.

Spre deosebire de defectele majore, cele minore nu afectează funcţionarea corectă a produsului. Exemple de astfel de defecte sunt: erori de ortografie în documente, poziţionarea incorectă a controalelor în programele de interfaţă cu utilizatorul etc.

Inspecţiile software sunt examinări stricte şi complete impuse de cerinţele, specificaţiile, arhitectura, design-ul, codul, planurile şi procedurile de testare ale produsului [Eben94]. Principalii indicatori de performanţă pentru fiecare produs prevăd criterii de terminare a fiecărei activităţi din ciclul său de dezvoltare. Aceşti indicatori includ completitudine, corectitudine, stil, reguli de construcţie şi viziuni multiple [Onei88].

Completitudinea se bazează pe transpunerea tuturor cerinţelor în cod, lucru esenţial pentru mentenanţă. Corectitudinea se bazează pe specificarea clară a funcţiilor şi transpunerea lor în cod, lucru esenţial pentru încrederea şi disponibilitatea produsului [Ling79]. Stilul se bazează pe consistenţa înregistrărilor, esenţială pentru mentenanţă. Regulile de construcţie se bazează pe arhitectura şi protocoalele aplicaţiei, şabloanele şi convenţiile utilizate pentru realizarea ei, lucru esenţial pentru disponibilitate şi încredere. Viziunile multiple se bazează pe o varietate de perspective şi puncte de vedere necesar a fi reflectate în produsul software, lucru esenţial pentru mentenanţă. Prin detectarea timpurie a defectelor şi prevenirea propagării lor în următoarele activităţi se elimină nevoia de evaluări şi a corecţiilor ulterioare, lucru esenţial pentru reducerea timpului şi costurilor de realizare.

Adoptarea inspecţiei software duce la creşterea competenţei şi este foarte apreciată de practicienii pregătiţi pentru utilizarea sa. Organizaţiile care folosesc această tehnică beneficiază de îmbunătăţirea predictibilităţii costurilor şi a performanţei proiectate, reducerea costurilor de dezvoltare şi întreţinere, reducerea defectelor şi creşterea satisfacţiei utilizatorilor.

Recuperarea investiţiei cu inspecţia software este definită ca raportul dintre economia netă şi costurile cu detecţia [Onei01b]. Economiile obţinute prin detectarea şi corectarea timpurie a defectelor previn creşterea costurilor care pot să apară dacă acestea ar fi detectate în etape ulterioare ale ciclului de dezvoltare a produsului. Un defect major nedetectat care se propagă în etapele ulterioare poate creşte costurile cu detecţia şi corectarea de două până la zece ori [Basi01]. Un defect minor poate creşte costurile cu detecţia şi corectarea de două până la patru ori. Economiile nete sunt deci de până la nouă ori în cazul defectelor majore şi de până la trei ori în cazul celor minore.

Inspecţia software este o activitate laborioasă, necesitând adesea evaluarea formală a codului, parcurgeri structurate şi alte tehnici similare. Rezultatele inspecţiilor software pot fi impresionante, dar adesea ele nu sunt bine realizate sau nu sunt aplicate de loc. Unii manageri consideră inspecţiile software ca fiind consumatoare de resurse, iar programatorii se simt adesea încorsetaţi în formalitatea procesului. Volumul total de cod

implicat este o altă constrângere, deoarece programele actuale, de o complexitate deosebită, implică adesea milioane de linii sursă. De aceea este unanim recunoscut faptul că inspecţia manuală se poate aplica efectiv numai pentru programe relativ simple.

Cu toate acestea, inspecţia software este cea mai eficientă metodă de evaluare şi garantează faptul că produsul final funcţionează conform specificaţiilor. În acelaşi timp este utilă şi proiectanţilor şi programatorilor care, pe viitor, vor fi în măsură să evite erorile constatate.

Pentru stabilirea componentelor de evaluat şi a metodelor ce se vor folosi trebuie avuţi în vedere următorii factori de risc [Onei01a]:

- componentele care folosesc tehnologii, tehnici sau instrumente noi;
- componentele critice;
- componentele ce folosesc algoritmi complecşi, care trebuie să fie corecţi şi optimizaţi;
- componentele cu multe condiţii de excepţie, care nu sunt uşor de testat;
- componentele care se vor reutiliza;
- componentele care vor servi drept model pentru alte componente;
- interfeţele cu utilizatorul;
- componentele realizate de dezvoltatori mai puţin experimentaţi.

Componentele care se încadrează în aceste categorii sunt considerate ca având un risc ridicat. Pentru acestea este necesară folosirea inspecţiei software. Componentele ale căror eventuale defecte neidentificate nu vor afecta semnificativ planul de realizare, calitatea, costurile şi obiectivele urmărite sunt considerate ca având un risc scăzut. Pentru evaluarea lor se pot folosi şi metode mai puţin formale.

34.3 Locul inspecţiei software în cadrul auditului informatic

Auditarea software constă în activităţi prin care se verifică gradul de concordanţă dintre specificaţii şi programul elaborat. Auditul software dă măsura siguranţei pe care trebuie să o aibă utilizatorul de programe atunci când obţine rezultate. Siguranţa se referă la corectitudinea şi completitudinea rezultatelor finale atunci când şi datele de intrare sunt corecte şi complete [Ivan05].

În urma verificărilor realizate se identifică diferenţe între cerinţe şi ceea ce s-a realizat, abateri faţă de standarde, norme şi restricţii impuse de tehnicile utilizate. În cazul cel mai fericit, se constată o suprapunere perfectă a acestora. Dacă diferenţele constatate nu sunt semnificative în raport cu criteriile de exigentă stabilite, produsul informatic este validat.

Inspecţia se realizează asupra tuturor produselor livrabile realizate de-a lungul ciclului de dezvoltare a aplicaţiilor informatice. În cadrul acestui proces, un rol special îl are inspecţia textului sursă, care constă în parcurgerea acestuia pas cu pas şi stabilirea concordanţei cu specificaţiile de programare.

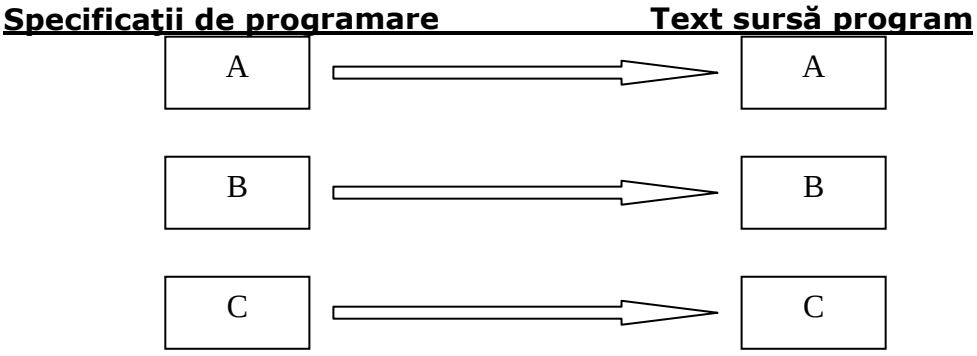
Pornind de la specificaţiile de programare unde se prezintă datele de intrare, rezultatele, modelele de calcul şi seturile de date de test şi

rezultatele ce trebuie obţinute, activitatea de inspecţie compară componentele specificaţiilor software cu secvenţele corespondente din program.

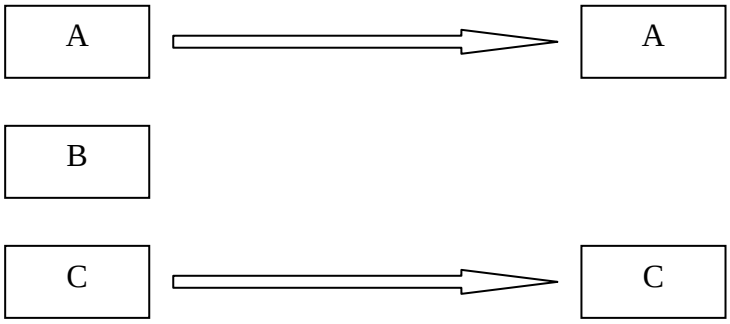
În urma inspecţiei se constată una din următoarele situaţii în care se pot afla specificaţiile de programare şi textul sursă:

a) tuturor secvenţelor de texte din specificaţii le corespund secvenţe de instrucţiuni în programul sursă. În acest caz programul realizează toate cerinţele definite în specificaţii;
b) numai anumitor secven□e de text din specificaţii le corespund secvenţe de instrucţiuni în programul sursă. În acest caz programul nu realizează toate cerinţele din specificaţii;
c) există mai multe secven□e de cod decât cele definite în specificaţii. Este cazul în care programatorul intuieşte o serie de prelucrări necesare programului, care nu au fost incluse în specificaţii;
d) există o diferenţă totală între specificaţii şi textul sursă. Programul realizează cu totul altceva decât s-a cerut prin specificaţii.

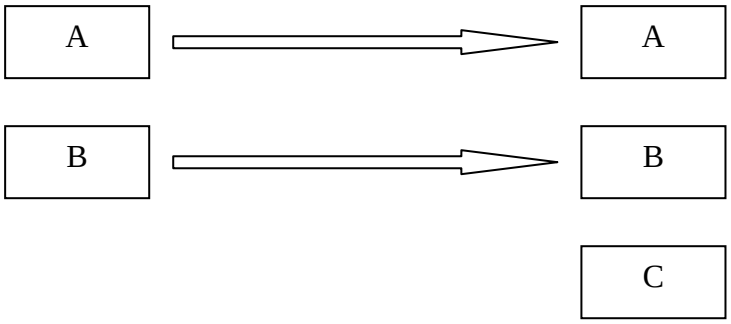
Toate aceste rezultate posibile ale comparării specificaţiilor de programare cu textul sursă al programelor sunt redate în figura 34.2.



a) specificaţiile sunt identice cu programul



b) programul realizează parţial cerinţele din specificaţii



c) Programul realizează mai multe funcţii decât au fost prevăzute în specificaţii

d) Programul face cu totul altceva decât a fost prevăzut în specificaţii

***Figura 34.2** Rezultatele posibile ale comparării specificaţiilor de programare cu textul sursă al programului*

În continuare se prezintă secvenţe din specificaţii împreună cu textul sursă aferent, realizat în limbajul C#, pentru exemplificarea cazurilor prezentate în figura 34.2.

a) Specificaţii identice cu programul corespund exemplului următor. Constructorul clasei PhotoManager este o metodă publică, fără parametri. Se iniţializează câmpul connection cu o nouă conexiune la SqlServer. Şirul de conectare se găseşte în fişierul de configurare al aplicaţiei şi este cel corespunzător bazei de date ”Personal”. Câmpul command se iniţializează cu o nouă comandă. Comanda se va executa pe conexiunea connection şi este de tip procedură stocată. Pe constructor se deschide conexiunea. Se iniţializează câmpul filter cu true sau false, în funcţie de rolul utilizatorului. Dacă utilizatorul este din grupul Administrators sau Friends, atunci filter devine false.

```
public PhotoManager() {
      Connection = new SqlConnection
(ConfigurationManager.ConnectionStrings[“Personal”].ConnectionString);
      command = new SqlCommand();
      command.Connection = conection;
      command.CommandType = CommandType.StoredProcedure;
      connection.Open();
      filter = true;
      if (HttpContext.Current.User.IsInRole(“Friends”) ||
HttpContext.Current.User.IsInRole(“Administrators”)) {
            Filter = false;
      }
}
```

b) Programul realizează parţial cerinţele din specificaţii. Operatorul de înmulţire a două obiecte de tip Matrice este o metodă publică şi statică din clasa utilitară Matrice. Numele predefinit este operator*. Parametrii sunt

două referinţe la obiecte de tip Matrice, m1 si m2. Metoda returnează rezultatul înmulţirii lui m1 cu m2 sub forma unei referinţe la obiect de tip Matrice alocat dinamic în corpul metodei. Se verifică dacă numărul de coloane al lui m1 este egal cu numărul de linii ale lui m2. Dacă este adevărat, se înmulţesc matricele, altfel se returnează valoarea *null*.

```
public static Matrice operator *(Matrice m1, Matrice m2)
      {
            Matrice c = null;
            c= new Matrice(m1.lin, m2.col);
            for (int i = 0; I < m1.lin; i++)
                  for (int j = 0; i < m2.col; j++)
                        for (int k = 0; k < m1.col; k++)
                              {
                              c[i,j] = c[i,j] + m1[i,k] * m2[k,j];
                              }
            return c;
      }
```

Aici, în cazul în care dimensiunile nu corespund se produce o excepţie care întrerupe funcţionarea execuţiei programului. Blocul cu alocare dinamică a lui c şi cele trei instrucţiuni for trebuie tratate într-un bloc al instrucţiunii

```
if ((m1.col == m2.lin))
      {…}
```

c) Programul realizează mai multe funcţii decât în specificaţii. GetPhoto este o metodă publică a clasei PhotoManager. Parametrii sunt photoid de tip int, reprezentând identificatorul imaginii în baza de date, şi dimensiunea size de tip int. Metoda returnează un obiect de tip Stream cu şirul de octeţi care alcătuieşte o imagine. Metoda apelează procedura stocată GetPhoto prin intermediul obiectului command. La comanda se adaugă parametrii procedurii, @PhotoID, @Size si @IsPublic, cu valorile, respectiv, photoid, size şi filter. Rezultatul execuţiei procedurii stocate se reţine în memorie într-un obiect result şi se construieşte obiectul stream pe baza acestuia.

```
public Stream GetPhoto(int photoid, int size) {
      command.CommandText = “GetPhoto”;
      command.Parameters.Add(new SqlParameter(“@PhotoID”, photoid));
      command.Parameters.Add(new SqlParameter(“@Size”, size));
      command.Parameters.Add(new SqlParameter(“@IsPublic”, filter));
      object result = command.ExecuteScalar();
      try {
            return new MemoryStream((byte[]result);
            } catch (ArgumentNullException e) {
                  Return null;
            }
      }
```

În acest exemplu, deşi nu este specificat în documentaţia primită, programatorul intuieşte că există posibilitatea ca obiectul returnat de procedura stocată să fie vid. Acest lucru ar produce în linia:

```
return new MemoryStream((byte[])result);
```

o excepţie ce ar întrerupe funcţionarea programului. De aceea el îmbracă codul cu o secţiune try – catch, tratând şi cazul omis de specificaţii. În cazul în care rezultatul este vid, programul nu se întrerupe, ci returnează un obiect vid.

În concluzie, auditul pe textul sursă analizează modul în care au fost introduse datele de test, ce proceduri au activat şi stabileşte caracterul parţial sau caracterul complet al prelucrărilor. Auditul pe textul sursă descoperă care sunt minusurile din textul sursă sau care sunt definirile în plus, fără a da soluţii, adică fără a genera secvenţele lipsă, pentru a arăta cum trebuie rescris programul.

Când se analizează secvenţele de program trebuie căutate sursele de erori care afectează fluxurile de prelucrare, dintre care se enumără [Ivan05]:

- lucrul cu variabile neiniţializate;
- traversarea unor structuri de date în afara limitelor pentru care au fost definite;
- neincluderea de teste care să evite împărţirea prin zero sau blocarea unor funcţii care au restricţii asupra intervalelor parametrilor;
- construirea unor expresii care filtrează parţial sau incorect submulţimi de elemente;
- restructurarea expresiilor prin schimbarea ordinii de evaluare a unor subexpresii diferite de cele date de specificaţii;
- efectuarea de conversii intermediare care deformează rezultatele finale;
- construirea expresiilor de referire a elementelor unei structuri care dezvoltă procese de căutare/regăsire neconcordante cu specificaţiile;
- absenţa manipulării variabilelor de stare a prelucrărilor;
- atribuirea de coduri variabilelor de stare după alte reguli decât cele date în specificaţiile de programe;
- alocarea dinamică a unor variabile fără a exista şi procesul de dealocare;
- introducerea unor elemente de neomogenitate, cum ar fi citiri scrieri de date;
- definirea de invarianţi în structuri repetitive în principal prin inexistenţa incrementărilor sau decrementărilor;
- neincluderea în secvenţe a unor instrucţiuni corespunzătoare evaluării unui model sau filtrării, ceea ce conduce la o tratare parţială a problemei;
- atribuirea altei semnificaţii variabilelor cu consecinţe directe asupra rolului şi locului pe care acestea le au în program;
- introducerea de expresii de atribuire care anulează prelucrările precedente;
- compunerea unor construcţii fără a exista o echivalenţă între formulele finale şi cele iniţiale ale acestora.

Inspecţia textului sursă presupune şi punerea în corespondenţă a variabilelor, ecuaţiilor şi inecuaţiilor, enumerărilor, definirilor de mulţimi şi intervale de existenţă conţinute în specificaţiile de programare şi în textul sursă.

În ceea ce priveşte variabilele, în cazul respectării riguroase a specificaţiilor se constată că lista variabilelor din specificaţii are aceeaşi dimensiune ca lista variabilelor din program (fiecărei variabile din specificaţii îi corespunde o variabilă şi numai una din program).

Dacă lista variabilelor din program este mai mare decât cea din specificaţii, trebuie analizată cauza acestei situaţii: i) în program sunt definite mai multe variabile intermediare decât este necesar, sau ii) specificaţiile sunt incomplete, caz în care trebuie identificate efectele asupra programului.

Dacă lista variabilelor din program este mai mică decât cea din specificaţii, acest lucru se poate datora faptului că i) programatorul a făcut redefiniri, caz ce nu constituie o eroare, sau ii) în program nu au fost abordate toate aspectele din specificaţii, ceea ce impune refacerea sa.

O ecuatie simplă din specificaţii trebuie să se regăsească în program sub forma unei instrucţiuni de atribuire:

- în specificaţii: a = b + c + d;
- în program: a = b + c + d.

O ecuaţie complexă trebuie să se regăsească sub forma unei structuri IF-ELSE-ENDIF sau a unei structuri CASE-ENDCASE, precum:

- în specificaţii:

$$a = \begin{cases} x^2 & \text{daca } x > 0 \\ \dfrac{1}{x^2} & \text{daca } x < 0 \\ 1 & \text{daca } x = 0 \end{cases} \qquad (34.1)$$

- în program:

```
if (b > 0) a = x * x;
else
       if (b < 0) a = 1 / (x * x);
       else
                    a = 1;
```

Cazul inecuaţiilor este similar cu cel al ecuaţiilor.

Indiferent de cele constatate, inspecţia textului sursă doar semnalează abaterile faţă de specificaţii, fără să dea soluţii. Va fi sarcina echipei de programare sa remedieze erorile identificate.

Un alt aspect pe care inspecţia trebuie să îl aibă în vedere este cel al structurilor de date. Programele lucrează cu fişiere, cu matrice, liste, stive, arbori binari, arbori B sau alte structuri de date agregate. Rolul inspecţiei software orientată pe structuri de date este de a vedea dacă:

- au fost bine alese structurile de date;
- implementarea procedurilor respectă cerinţele impuse de rezolvarea problemei;
- complexitatea structurilor de date utilizate este în concordanţă cu complexitatea problemei;
- expresiile de referire a elementelor din structuri sunt cât mai simple posibil.

Pentru aceasta trebuie analizate tipurile de structuri folosite. În cazul în care se cunoaşte numărul componentelor care alcătuiesc colectivitatea, trebuie să se lucreze cu fişiere. În caz contrar, când nu se cunoaşte numărul

componentelor care alcătuiesc colectivitatea, trebuie să se lucreze cu structuri dinamice. Inspectorul analizează ipotezele din specificaţii şi verifică dacă acest lucru este respectat. Inspecţia are rolul de a evidenţia dacă structura de date folosită este cea adecvată. În cazul în care aceasta nu este cea adecvată, se va demonstra cu calcule care este structura cea mai potrivită.

O cerinţă fundamentală a prelucrării datelor este aceea de a memora date aleatoare. Dacă un şir de valori este constant, atunci trebuie să se memoreze numărul de elemente şi valoarea constantelor. Dacă şirul are elemente ce rezultă din calcul, atunci se memorează valoarea de start, expresia analitică şi numărul de termeni. Pentru aceasta se analizează zona de memorie cu informaţia utilă în care se descriu caracteristicile elementelor colectivităţii pentru care se efectuează prelucrarea, zonele unde se definesc legăturile dintre elemente şi care sunt variabilele pointer.

Pentru structurile de date existente în programme se defineşte volumul de prelucrări legat de numărul de comparări necesare pentru a ajunge la elementul căutat din structură. Pentru aceasta, inspectorul analizează structurile definite în program, estimează volumul necesar de comparări şi verifică dacă structura de date utilizată este cea care determină volumul cel mai redus de comparări.

34.4 Structura echipei de inspecţie software

Când dezvoltatorii îşi revizuiesc propria muncă nu observă toate defectele produsului. De aceea este necesară revizuirea produsului de către o nouă echipă.

Echipa de inspecţie software este formată dintr-un grup de oameni care lucrează împreună pentru a analiza fiecare produs al unei activităţi de dezvoltare, cu scopul de a identifica şi înlătura defectele. Acest lucru este realizat prin atribuirea de cinci roluri procedurale diferite pentru membrii echipei: autor, moderator, lector (reader), înregistrator (recorder) şi inspector [Fran**].

Autorul este persoana care a realizat produsul sau un stadiu al acestuia şi este ultimul responsabil cu actualizarea acestuia după inspecţia software. De regulă este persoana care a creat iniţial produsul. Lui i se vor adresa întrebări specifice cu privire la conţinutul produsului şi îşi va folosi cunoştinţele speciale pentru a ajuta la detectarea defectelor reale şi pentru a preveni ca simple neînţelegeri să fie considerate defecte. El va corecta toate defectele majore identificate precum şi, în limita timpului şi resurselor disponibile, defectele minore.

Moderatorul conduce echipa de inspecţie software şi este responsabil cu asigurarea faptului că procedurile inspecţiei sunt respectate de-a lungul întregului proces de inspecţie. Aceasta include asigurarea că toţi membrii echipei îşi îndeplinesc rolurile la nivelul sarcinilor. Sarcinile sale sunt:

- verificarea faptului că produsul este pregătit pentru inspecţia software;
- verificarea îndeplinirii criteriilor de început;
- alegerea efectivă a echipei de inspecţie software;
- înregistrarea întâlnirilor de inspecţie software;
- verificarea îndeplinirii criteriilor de terminare.

Moderatorul este implicat activ în toate etapele inspecţiei software, mai puţin în cea de refacere.

Lectorul este responsabil cu conducerea echipei în timpul întâlnirilor de inspecţie software, prin prezentarea unor unităţi logice mici.

Înregistratorul documentează toate defectele identificate în timpul întâlnirilor de inspecţie software. Această documentaţie include locul unde a fost identificat fiecare defect, o scurtă descriere a sa, precum şi numele inspectorului care l-a găsit. În plus, fiecare defect este asociat unei categorii şi unui tip de defecte. Toate defectele sunt înregistrate într-o listă de defecte.

Independent de rolul atribuit, toţi membrii echipei au şi rolul de *inspectori*. Rolul de inspector este responsabil cu analiza produsului şi identificarea defectelor sale. În funcţie de aspectele urmărite, inspectorii pot avea următoarele roluri [Dona**]:

- *arhitect de software*, cu rolul de a inspecta proiectul din punct de vedere al calităţii şi conformităţii cu arhitectura software;
- *programator*, cu rolul de a inspecta componentele software din punct de vedere al conformităţii cu standardele de codificare;
- *inginer de testare*, cu rolul de a inspecta produsul software pe componente şi integrat;
- *inginer de calitate*, cu rolul de a inspecta componentele software din punct de vedere al calităţii şi conformităţii cu convenţiile utilizate;
- *inginer de securitate*, cu rolul de a inspecta componentele software din punct de vedere al cerinţelor şi mecanismelor de securitate;
- *reprezentantul utilizatorilor*, cu rolul de a inspecta interfaţa cu utilizatorul din punct de vedere al facilităţilor de utilizare.

În funcţie de pregătirea pe care o are, un inspector poate îndeplini unul sau mai multe din aceste roluri. Important este ca un anumit aspect să fie urmărit de o singură persoană, să nu existe suprapuneri în activitatea de inspecţie şi, în acelaşi timp, echipa să nu fie prea numeroasă.

Fiecare inspector va întocmi un raport de inspecţie cu cele constatate. Aceste rapoarte vor sta la baza întocmirii raportului final, care va concluziona cele constatate de întreaga echipă.

Echipa realizează inspecţia software pe parcursul unui set de etape predefinite, a căror înlănţuire este prezentată în figura 34.3.

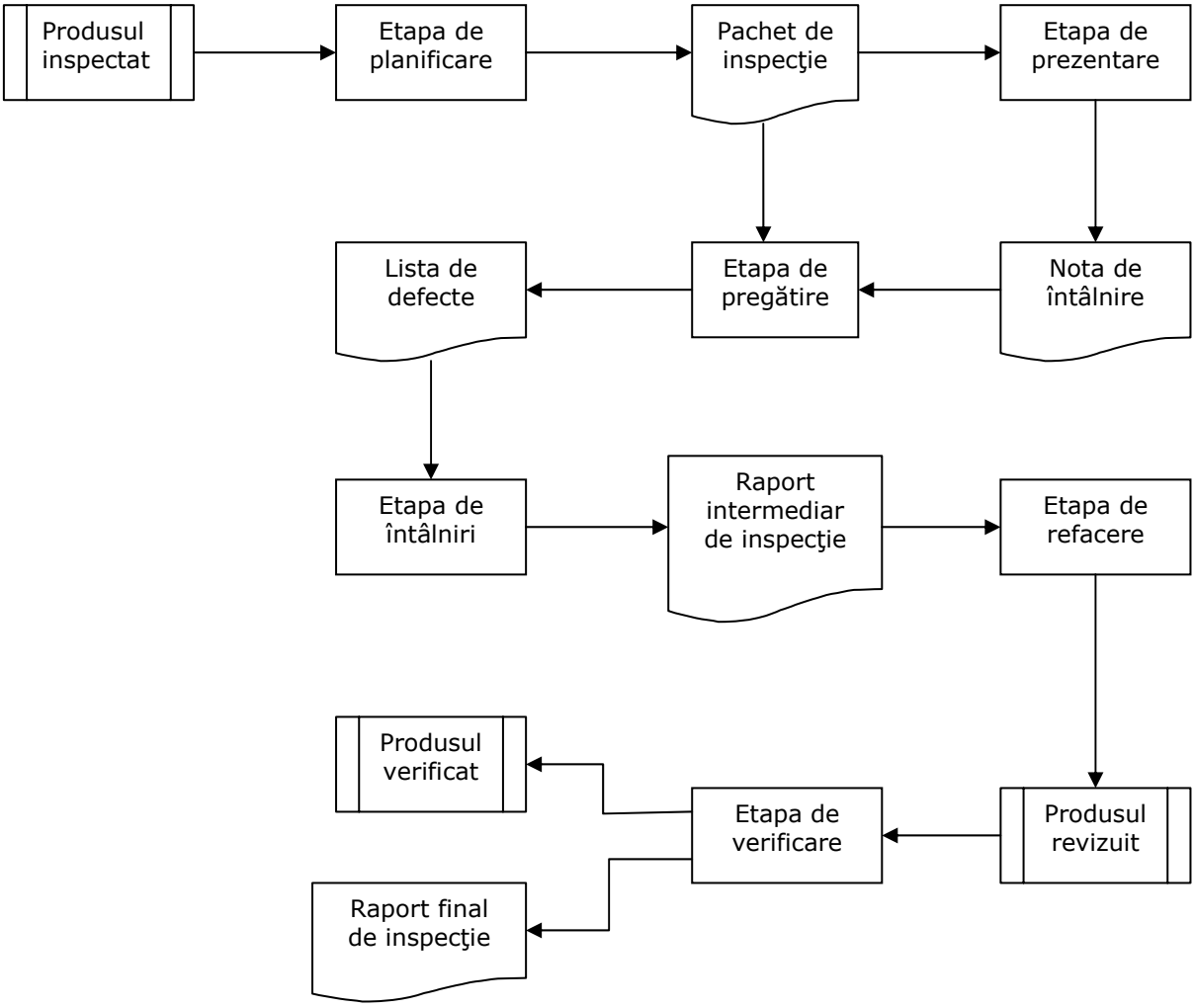


Figura 34.3 *Etapele inspecţiei software.*

Pentru unele produse software se defineşte un singur tip de inspecţie, în timp ce pentru alte produse sunt necesare mai multe tipuri de inspecţie. Un produs software are tipuri diferite de inspecţie pentru specificaţii, design, cod sursă. Fiecare tip de inspecţie este unic şi complet definit de şapte parametrii, prezentaţi în figura 34.4.

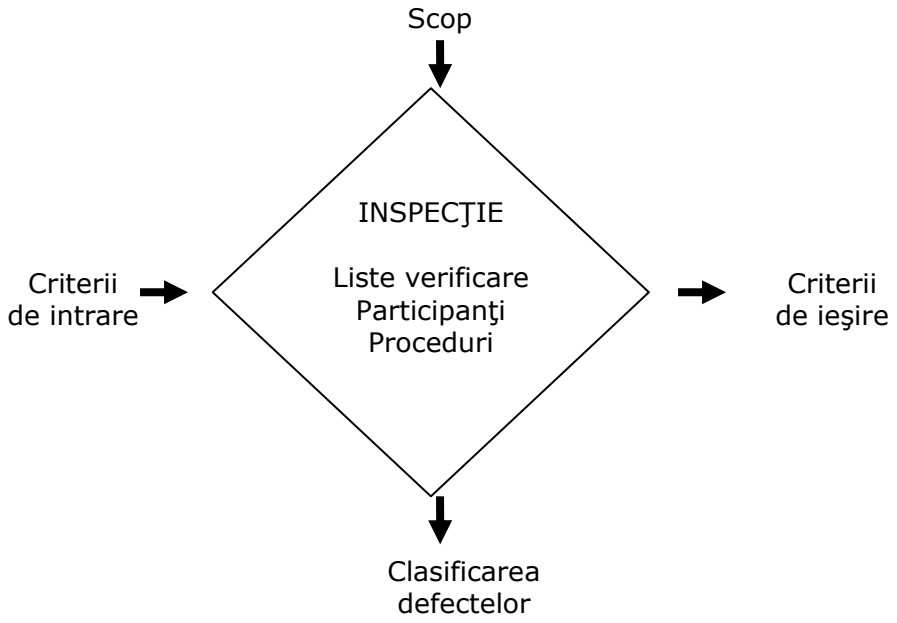


Figura 34.4 *Parametrii inspecţiei*

Scopul descrie ceea ce va realiza inspecţia. Sopul general al inspecţiei este verificarea produsului software.

Criteriile de intrare se referă la datele de intrare pentru etapa de întrunire. Acestea includ produsul software asupra căruia se va realiza inspecţia, cerinţele şi specificaţiile produsului, precum şi alte rapoarte necesare. Criteriile minime de intrare:

- codul produsului – codul a fost compilat cu succes, fară mesaje de eroare;
- documentaţia produsului – ortografie corectă.

Listele de verificare conţin indicaţii şi recomandări adresate inspectorilor pentru a găsi cu uşurinţă defectele produsului analizat.

Participanţii sunt reprezentaţi de persoanele care vor îndeplini procesul de inspecţie.

Procedurile explică cum trebuie abordat fiecare tip de inspecţie.

Clasificarea defectelor descrie ceea ce trebuie asociat cu un defect şi cum se înregistrează acestea. De obicei există trei caracteristici care sunt atribuite unui defect: tipul, circumstanţa şi gradul de impact. Aceste caracteristici sunt denumite tipul defectului, clasa şi gravitatea. Împreună acestea ajută la procesul de identificare a defectelor şi la transmiterea rezultatelor inspecţiei.

Criteriile de ieşire indică inspectorilor momentul în care procesul de inspecţie este terminat. Cel mai întâlnit criteriu de ieşire este faptul că toate defectele identificate în etapa de întrunire sunt corectate. Criteriile de ieşire includ, de obicei, produsul software verificat şi corectat, precum şi un raport de inspecţie.

Scopul principal al procesului de inspecţie software este de a elimina defectele dintr-un anumit produs. Produsul poate fi: specificarea cerinţelor, manualul de proiectare sau module program. Pentru a determina dacă un produs este pregătit pentru inspecţie se folosesc anumite criterii de intrare. Sfârşitul inspecţiei se realizează la atingerea criteriilor de terminare a procesului. Criteriile de intrare depind de fiecare produs în parte, dar în

general se referă la a determina dacă produsul este suficient de matur pentru a fi folosit după înlăturarea defectelor. În general, criteriile de ieşire trebuie să asigure că au fost înlăturate toate defectele identificate în timpul procesului de inspecţie. Este posibil ca dintre aceste criterii să se excludă corectarea defectelor minore, care nu au impact negativ în utilizarea produsului.

Indiferent ce este produsul analizat, procesul de inspecţie va urma aceleaşi etape. Aceste etape sunt prezentate în tabelul 34.1, [Fran**].

Tabelul nr. 34.1 *Etapele procesului de inspecţie*

Etapa	Descriere
Planificare	Identificarea produsului de inspectat şi stabilirea planului de inspecţie
Prezentare	Etapă opţională în care membrii echipei care nu sunt familiarizaţi cu produsul de inspectat se familiarizează cu problema
Pregătire	Membrii echipei inspectează individual produsul pentru găsirea defectelor
Întâlniri	Membrii echipei se întâlnesc pentru a discuta posibilele defecte găsite
Refacere	Produsul este revizuit conform cerinţelor şi specificaţiilor
Verificare	Produsul refăcut este verificat, datele finale ale inspecţiei sunt colectate şi centralizate şi procesul de inspecţie este închis

34.5 Planificarea inspecţiei software

Există părerea că inspecţiile sunt realizate în ultimul moment astfel încât produsul poate fi considerat terminat şi livrat clientului sau trecut în următoarea etapă de dezvoltare. Nu aceasta este realitatea atâta timp cât inspecţiile trebuie planificate cu grijă şi membrii echipei trebuie coordonaţi pentru atingerea scopului comun.

Primul pas în iniţierea unei inspecţii apare atunci când autorul consideră că produsul îndeplineşte toate condiţiile de intrare în etapa de planificare. Autorul notifică acest lucru moderatorului şi este responsabilitatea acestuia să verifice îndeplinirea condiţiilor. Criteriile de intrare specifică materialele ce vor fi inspectate şi condiţiile pe care acestea trebuie să le îndeplinească. Dacă aceste criterii nu sunt îndeplinite, moderatorul informează autorul despre ce mai trebuie făcut pentru satisfacerea lor. Această verificare este importantă pentru procesul de inspecţie deoarece oferă o garanţie că produsul se află într-o etapă potrivită pentru începerea inspecţiei.

După verificarea criteriilor de intrare, moderatorul selectează membrii echipei şi le atribuie roluri. După alegerea acestora, moderatorul poate determina dacă inspectorii deţin suficiente informaţii despre produs pentru îndeplinirea rolurilor atribuite. Dacă acest lucru nu este realizat, se va programa o întâlnire de prezentare pentru ca inspectorii să primească toate informaţiile necesare.

Scopul principal al etapei de planificare este de a asigura eficienţa necesară procesului de inspecţie. Pentru îndeplinirea acestui obiectiv, în etapa de planificare trebuie avute în vedere următoarele aspecte:

- echipa de inspecţie va fi formată din următoarele persoane: moderator, înregistrator, inspector, lector şi autor;
- produsul ce urmează a fi inspectat va fi "îngheţat" pe durata procesului de inspecţie;
- întâlnirile echipei nu vor depăşi două ore;
- managerii de proiect vor identifica punctele de verificare în planul proiectului pentru conducerea inspecţiei;
- lista de verificare va fi folosită de inspectori pentru a ajuta la identificarea defectelor;
- trebuie asigurate cel puţin două ore de pregătire;
- la un moment dat se vor inspecta maximum 20 pagini ale produsului;
- autorul nu va îndeplini rolul de moderator, lector sau înregistrator.

În tabelul 34.2 se prezintă scopul, activităţile şi rolurile procedurale ale etapei de planificare a inspecţiei software.

Tabelul nr. 34.2 *Activităţile şi rolurile etapei de planificare*

Scop	▪ Organizarea şi planificarea resurselor;
Intrări	▪ Forma finală a produsului; ▪ Materiale suport; ▪ Criterii de demarare a inspecţiei; ▪ Datele de inspectat (dacă sunt disponibile);
Sarcini	▪ Verificarea criteriilor de demarare; ▪ Selectarea echipei; ▪ Determinarea necesităţii unei prezentări; ▪ Întocmirea planului de inspecţie; ▪ Completarea şi distribuirea pachetului de inspecţie;
Măsurători	▪ Planificarea efortului; ▪ Dimensiunea produsului;
Ieşiri	▪ Definitivarea şi distribuirea pachetului de inspecţie ▪ Planificarea unei prezentări (opţional).
Roluri	Moderatorul: ✓ verifică îndeplinirea criteriilor de demarare; ✓ selectează membrii echipei de inspecţie; ✓ decide dacă este necesară organizarea unei prezentări; ✓ planifică întâlnirile şi întocmeşte notele de întâlnire; Autorul: ✓ revede produsul cu moderatorul; ✓ propune membrii echipei; ✓ recomandă, dacă este cazul, o întâlnire de prezentare;

34.6 Etapa de prezentare

Obiectivul tuturor inspecţiilor software este de a analiza produsul şi de a detecta şi înlătura defectele. Pentru aceasta este necesar un anumit nivel de cunoaştere a produsului. Dacă moderatorul consideră că inspectorii nu deţin suficiente cunoştinţe, poate organiza o întâlnire de prezentare pentru punerea lor în temă. Autorul va explica funcţionalitatea produsului şi va face o descriere a tuturor tehnicilor şi convenţiilor folosite.

În tabelul 34.3 se prezintă scopul, activităţile şi rolurile procedurale ale etapei de prezentare.

Tabelul nr. 34.3 *Activităţile şi rolurile etapei de prezentare*

Scop	▪ Informare;
Intrări	▪ Produsul de inspectat; ▪ Prezentarea materialului;
Sarcini	▪ Moderatorul conduce întâlnirea de prezentare; ▪ Autorul prezintă produsul;
Măsurători	▪ Timpul de pregătire necesar autorului; ▪ Timpul necesar întâlnirii;
Ieşiri	▪ Înţelegerea mai bună a produsului; ▪ Atribuirea responsabilităţilor; ▪ Pachetul de inspecţie a produsului; ▪ Materialul întâlnirii de prezentare; ▪ Nota de întâlnire;
Roluri	Moderatorul: ✓ conduce întâlnirea astfel încât să maximizeze schimbul de informaţii; ✓ dacă este necesar, atribuie responsabilităţi (teste, standarde, claritate) pentru fiecare inspector; Autorul: ✓ pregăteşte întâlnirea; ✓ prezintă materialul; ✓ răspunde la întrebări; Inspectorul: ✓ pune întrebări pentru o mai bună înţelegere a produsului; ✓ nu revizuieşte şi nu discută alternative de proiectare/implementare;

34.7 Etapa de pregătire a inspecţiei

Aceasta este cea mai importantă etapă a procesului de inspecţie. În timpul etapei de pregătire, fiecare membru al echipei examinează individual produsul, căutându-i defectele. Pentru o mai bună analiză a produsului, fiecărui inspector îi sunt atribuite anumite categorii de defecte.

De notat că defectele nu apar doar în etapa de codificare a proiectului. Cele mai multe produse software trec prin mai multe etape şi produc diferite produse. De câte ori rezultă un produs există posibilitatea de

apariţie a unor defecte. Pentru fiecare din aceste produse procesul de inspecţie trebuie să prevadă o listă de verificare care descrie defectele fazei respective. În tabelul 34.4 se prezintă un exemplu cu cerinţele listei de verificare.

Tabelul nr. 34.4 *Cerinţele listei de verificare*

Categoria de defecte	**Întrebări**
Claritate	▪ Cerinţele sunt clare si neambigue?
Standarde	▪ Toate cerinţele au standarde ce trebuie urmărite?
Completitudine	▪ Toate cerinţele sunt complete?
Nivelul de detaliu	▪ Cerinţele sunt independente de proiectare?
Consistenţă	▪ Cerinţele sunt consistente? ▪ Datele sunt structurate şi funcţiile au nume şi sunt folosite consistent?
Funcţionalitate	▪ Funcţiile sunt corect specificate? ▪ Intrările şi ieşirile sunt specificate clar? ▪ Funcţiile sunt logic independente şi formează o structură bine organizată?
Performanţă	▪ Sunt clar definite cerinţele de performanţă pentru sincronizare, folosirea memoriei şi a resurselor?
Testabilitate	▪ Există cerinţe pentru testare şi verificare?
Fezabilitate	▪ Fiecare componentă poate fi implementată cu tehnicile, instrumentele, resursele şi personalul disponibil şi cu restricţiile de costuri şi planificare specificate?
Maleabilitate	▪ Cerinţele sunt unic identificate?
Capacitate de modificare	▪ Cerinţele sunt unic structurate astfel încât orice modificare necesară să poată fi făcută uşor, complet şi consistent?

Toate defectele posibile identificate de inspectori în etapa de pregătire trebuie înregistrate. De notat că în acest moment acestea nu vor fi considerate defecte ale produsului. Fiecare inspector va înregistra potenţialele defectele găsite, dar numai în etapa de întâlnire se va stabili care dintre ele sunt defecte reale şi care nu.

În tabelul 34.5 se prezintă scopul, activităţile şi rolurile procedurale ale etapei de pregătire.

Tabelul nr. 34.5 *Activităţile şi rolurile etapei de pregătire*

Scop	▪ Înţelegerea produsului şi identificarea potenţialelor defecte;
Intrări	▪ Pachetul de inspecţie; ▪ Înţelegerea produsului; ▪ Materialul revizuit al inspecţiei;
Sarcini	▪ Studierea produsului; ▪ Identificarea defectelor:

	✓ Dacă este necesară o inspecţie a cerinţelor, se foloseşte lista de inspecţie a cerinţelor software; ✓ Dacă este necesară o inspecţie la nivelul proiectării de detaliu, se foloseşte lista de inspecţie a proiectării detaliate; ✓ Dacă este necesară inspecţia codului, se determină dacă există o listă pentru limbajul în care este scris codul. Dacă există, se foloseşte lista respectivă. Dacă nu, se adaptează lista principală pentru inspecţia codului la cerinţele limbajului de inspectat; ▪ Înregistrarea timpului de pregătire; ▪ Completarea listei de defecte;
Măsurători	▪ Timpul de pregătire; ▪ Numărul de defecte;
Ieşiri	▪ Lista completă de defecte; ▪ Înregistrarea timpului de pregătire;
Roluri	Inspectorul: ✓ studiază produsul; ✓ identifică defectele; ✓ înregistrează timpul de pregătire; Moderatorul: ✓ realizează toate sarcinile de pregătire a inspecţiei; Lectorul: ✓ stabileşte cum să prezinte produsul;

34.8 Etapa de întâlnire

În etapa de întâlnire întreaga echipă este responsabilă cu identificarea defectelor produsului. Toţi membrii echipei participă la aceste întâlniri şi prezintă defectele identificate în etapa de pregătire.

În primul rând moderatorul revizuieşte agenda întâlnirii, prezintă participanţii şi rolurile lor. Apoi lectorul prezintă inspectorilor produsul, împărţit în mici unităţi logice (de exemplu: paragrafe de text, blocuri de cod, etc.). Pe măsura prezentării unităţilor logice, fiecare inspector caută defectele din unitatea respectivă. Când sunt găsite posibile defecte, întregul grup discută dacă reprezintă sau nu defecte adevărate.

Este necesară realizarea unui consens în ceea ce priveşte fiecare defect, deoarece uneori este posibil ca un potenţial defect să fie o greşeală din partea inspectorului sau o neînţelegere care poate fi clarificată de autor. Când se realizează consensul în ceea ce priveşte un defect, lectorul îl trece în lista de defecte într-o manieră clară şi concisă. Este important ca moderatorul să menţină atenţia participanţilor asupra detectării defectelor şi să nu lase discuţia să devieze spre încercarea de a determina modul de corecţie a lor. Refacerea produsului se va realiza de către autor după întâlnirea de inspecţie. Lista de defecte are rolul de a ajuta autorul să refacă produsul şi este folosită în etapele ulterioare ale procesului de inspecţie

pentru a se verifica că toate defectele identificate la întâlnirea de inspecţie au fost corectate.

Pentru folosirea judicioasă a timpului, o întâlnire de inspecţie trebuie să urmeze o agendă de lucru predefinită. Se recomandă ca agenda să conţină următoarele capitole:

1. Introducere
2. Stabilirea pregătirii inspectorilor
3. Citirea produsului, identificarea şi înregistrarea defectelor
4. Analiza defectelor
5. Determinarea dispoziţiei produsului

Echipa de inspecţie foloseşte lista de defecte pentru a determina dispoziţia produsului. Dispoziţia se referă la procedura de folosit pentru verificarea refacerii produsului de către autor. Robert Ebenau [Eben94] propune următoarele trei categorii de dispoziţii:

a) *dispoziţie de tip A:* acceptarea produsului ca fiind complet, fără nici o altă verificare a refacerii sale. Aceasta nu presupune că produsul nu mai are nici un defect, ci că nu există defecte care să determine abaterea de la specificaţiile sale şi că sunt doar câteva defecte minore care pot fi lăsate la latitudinea autorului.
b) *dispoziţie de tip C:* acceptarea condiţionată a produsului, cu verificarea refacerii de către moderator, împreună cu autorul. Aceasta este situaţia când există unele defecte majore, puţine la număr, iar corectarea lor nu va duce la modificări majore în premisele de proiectare ale produsului.
c) *dispoziţie de tip R:* reinspectarea produsului refăcut de autor. Aceasta presupune ca produsul refăcut să fie examinat de către moderator, autor şi cel puţin un membru al echipei de inspecţie într-o nouă întâlnire de inspecţie. Acesta este cazul în care există un număr însemnat de defecte majore, sau când refacerea va modifica substanţial premisele de proiectare ale produsului.

În tabelul 34.6 se prezintă scopul, activităţile şi rolurile procedurale ale etapei de întâlnire.

Tabelul nr. 34.6 *Activităţile şi rolurile etapei de întâlnire*

Scop	▪ Atingerea consensului privind defectele; verificarea produsului;
Intrări	▪ Lista completă cu defecte; ▪ Timpul de pregătire înregistrat;
Sarcini	▪ Introducere; ▪ Citirea produsului; ▪ Identificarea şi înregistrarea defectelor; ▪ Rezumatul revizuirii defectelor; ▪ Determinarea dispoziţiei produsului;
Măsurători	▪ Gradul de corectitudine a datelor; ▪ Datele defecte;
Ieşiri	▪ Dispoziţia produsului; ▪ Lista completă de defecte; ▪ Sumarul complet al defectelor; ▪ Nota completă a întâlnirii de inspecţie; ▪ Raportul parţial de inspecţie;

Roluri	Autorul: ✓ rămâne un ascultător obiectiv şi activ, ia notiţe; ✓ nu adoptă o atitudine defensivă; Inspectorii: ✓ obiectivi şi impersonali; ✓ bine pregătiţi şi participanţi activi; ✓ nu trebuie să ridice probleme de stil şi soluţii de dezvoltare; Moderatorul: ✓ începe şi termină şedinţa la timp, menţinând o atmosferă adecvată; ✓ urmăreşte menţinerea sinergiei întâlnirii; ✓ se asigură că defectele întrunesc consensul echipei; Lectorul: ✓ stabileşte cea mai bună strategie de prezentare a produsului; ✓ răspunde la întrebările puse de membrii echipei; Înregistratorul: ✓ trebuie să fie familiarizat cu schema de clasificare a defectelor; ✓ înregistrează toate datele în lista principală de defecte; ✓ sintetizează datele defecte în sumarul de inspecţie;

34.9 Etapa de refacere

Defectele identificate în etapa de întâlnire sunt revăzute de autor în etapa de refacere. Autorul foloseşte lista de defecte pentru a determina care sunt acestea şi unde sunt ele localizate. Când toate defectele sunt rezolvate, autorul prezintă din nou produsul moderatorului pentru următoarea etapă de inspecţie.

Există situaţii în care un defect identificat de echipa de inspecţie nu este înlăturat în această etapă. Oricare ar fi motivul pentru care corecţia este amânată, decizia pentru acest lucru trebuie să aparţină managerului de proiect. Defectul va fi notat în sistemul de control al modificărilor proiectului şi va fi urmărit de managerul de proiect până când va fi posibil de corectat efectiv.

În tabelul 34.7 se prezintă scopul, activităţile şi rolurile procedurale din etapa de refacere.

Tabelul nr. 34.7 *Activităţile şi rolurile etapei de refacere*

Scop	▪ Corectarea şi rezolvarea defectelor;
Intrări	▪ Lista completă cu defecte; ▪ Timpul de pregătire înregistrat;

Sarcini	Autorul: ✓ rezolvă toate defectele; ✓ notifică moderatorului terminarea refacerii; ✓ raportează managerului de proiect defectele nerezolvate;
Măsurători	▪ Numărul de defecte; ▪ Efortul de refacere;
Ieşiri	▪ Defecte rezolvate; ▪ Înregistrarea timpului de refacere în lista principală de defecte; ▪ Notificarea către moderator a terminării refacerii; ▪ Produsul revizuit;
Roluri	Autorul: ✓ rezolvă defectele; ✓ raportează managerului de proiect defectele nerezolvate; ✓ notifică moderatorului terminarea refacerii;

34.10 Etapa de verificare a refacerii

Toate revizuirile produsului făcute de autor în etapa de refacere trebuie verificate formal, deoarece este posibil ca autorul să fi provocat noi defecte în încercarea de corectare a defectelor identificate anterior. În funcţie de dispoziţia atribuită produsului în timpul întâlnirilor de inspecţie, moderatorul decide:

- pentru dispoziţie de tip C (acceptare condiţionată) moderatorul va verifica reviziile făcute. El se va concentra atât pe conţinutul reviziilor, cât şi pe interfaţa acestora cu restul documentului.
- pentru dispoziţie de tip R (reinspectarea produsului) se va relua întreg procesul de inspecţie, dar cu focalizarea pe reviziile făcute şi pe interdependenţele dintre ele.

După ce autorul realizează toate modificările cerute şi moderatorul verifică reviziile, acesta completează celelalte rapoarte de inspecţie.

În tabelul 34.8 se prezintă scopul, activităţile şi rolurile procedurale din etapa de verificare.

Tabelul nr. 34.8 *Activităţile şi rolurile etapei de verificare*

Scop	▪ Terminarea şi certificarea inspecţiei;
Intrări	▪ Produsul revizuit, dacă dispoziţia sa a fost de tip C; ▪ Produsul, dacă dispoziţia sa a fost de tip A; ▪ Toate situaţiile de inspecţie;
Sarcini	▪ Moderatorul verifică produsul: ✓ dacă dispoziţia produsului a fost de tip R, se reia de la etapa de planificare; ✓ dacă dispoziţia produsului a fost de tip C, se verifică refacerea şi, dacă este necesar, se reia etapa de refacere; ✓ dacă dispoziţia produsului a fost de tip A, se certifică inspecţia;

	▪ Moderatorul completează raportul de concluzii; ▪ Moderatorul transmite managerului de proiect fişierul de inspecţie;
Măsurători	▪ Efortul de verificare; ▪ Raportul de concluzii;
Ieşiri	▪ Produsul verificat; ▪ Certificarea inspecţiei; ▪ Raportul complet de concluzii; ▪ Fişierul de inspecţie;
Roluri	Autorul: ✓ planifică reinspecţia, dacă dispoziţia produsului a fost de tip R; ✓ realizează şi reface ceea ce a identificat moderatorul; Moderatorul: ✓ întocmeşte planul reinspecţiei, dacă dispoziţia produsului a fost de tip R; ✓ examinează refacerea împreună cu autorul, dacă dispoziţia produsului a fost de tip C; ✓ completează raportul de concluzii;

34.11 Elaborarea rapoartelor de inspecţie

Sunt patru rapoarte standard care trebuie folosite când este programată o întâlnire de inspecţie, când sunt identificate defecte sau când sunt pregătite rapoarte pentru manageri. În timpul procesului de inspecţie, aceste rapoarte constituie un mecanism de comunicare între membrii echipei şi manageri. Cele patru rapoarte standard, al căror flux este prezentat în figura 34.5, sunt: nota de întâlnire, lista de defecte, raportul de concluzii şi raportul pentru management.

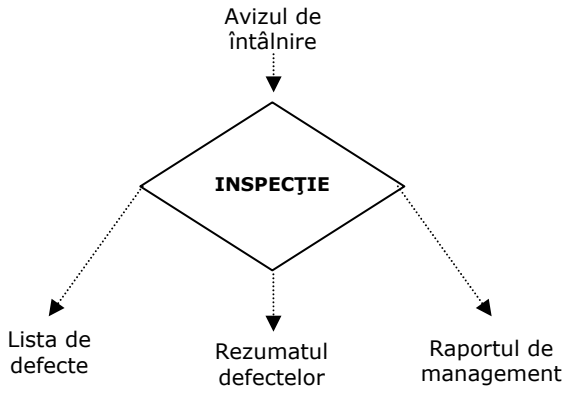


Figura 34.5 *Fluxul rapoartelor inspecţiei.*

Nota de întâlnire este întocmită de moderator în etapa de planificare şi scopul său este de a informa membrii echipei despre inspecţia care urmează. De obicei raportul este trimis membrilor echipei ca parte a pachetului de lucru, care include toate materialele necesare pentru

demararea etapei de planificare. Pachetul poate include produsul, nota de întâlnire, lista de defecte şi alte liste de control.

Nota de întâlnire trebuie să conţină următoarele informaţii ce vor fi completate de moderator înainte de a fi transmise membrilor echipei:

- antetul notei – informaţii administrative care includ: numele proiectului, data distribuirii, activitatea, componenta, versiunea, documentul, numele moderatorului;
- tipul întâlnirii – inspecţie, reinspecţie, mentenanţă;
- tipul inspecţiei – cerinţe, proiectare de ansamblu, proiectare de detaliu, cod;
- programul întâlnirii – data, durata, timpul de pregătire, dimensiunea produsului;
- membrii echipei de inspecţie – numele, categoriile de defecte pentru care sunt responsabili, rolurile procedurale;
- comentarii.

Lista de defecte este raportul folosit de inspectori în etapa de pregătire. Raportul conţine un antet şi secţiuni pe tipuri de întâlniri, la fel ca nota de întâlnire descrisă anterior. Lista de defecte conţine următoarele informaţii care ajută la înregistrarea defectelor identificate:

- *locaţia* – este completată de inspectori în etapa de pregătire; această zonă este folosită pentru localizarea defectelor şi conţine:
  - numărul defectului: este numărul de identificare atribuit defectului;
  - numărul de pagină: este numărul paginii produsului unde a fost identificat defectul;
  - numărul de secţiune: este numărul secţiunii produsului unde a fost identificat defectul;
  - numărul de paragraf: este numărul paragrafului unde a fost identificat defectul;
  - numărul de propoziţie: este numărul propoziţiei unde a fost identificat defectul;
- *descrierea defectului* – este completată de inspectori în etapa de pregătire, într-o formă scurtă şi concisă.
- *defectul* – această zonă este completată de moderator în etapa de întâlnire. Este folosită pentru clasificarea defectelor şi se compune din:
  - acceptare D/N: se bifeaza "D" dacă există consensul inspectorilor că defectul identificat reprezintă un defect real;
  - tipul defectului: este un număr care identifică categoria de defecte din lista de control a inspectorilor;
  - clasa defectului: echipa de inspecţie poate decide că defectul a fost cauzat de ceva care lipseşte, este eronat sau este din afara produsului; suplimentar, echipa poate preciza "nu este sigur";
  - gravitatea defectului: se specifică dacă defectul este considerat unul major, mediu sau minor, corespunzător impactului potenţial asupra produsului dacă nu este înlăturat;
- *timpul de refacere* – este completat de autor în etapa de refacere şi reprezintă timpul necesar pentru remedierea defectului.

Raportul de concluzii este completat după etapa de întâlniri. El sumarizează tipurile, clasele şi gravitatea tuturor defectelor identificate în

timpul întâlnirilor de inspecţie şi conţine un antet şi secţiuni pe tipuri de întâlniri, la fel ca în raportul de întâlniri descris anterior.

Raportul trebuie să conţină, pentru fiecare categorie de defecte (majore, medii şi minore), descrierea consecinţelor, prin specificarea clasei de defecte: lipsă, eronat, din afara produsului, nu este sigur.

Raportul de management este completat de moderator la sfârşitul etapei de întâlniri. Conţine un antet şi secţiuni pe tipuri de întâlniri, la fel ca la raportul de întâlniri prezentat anterior. Raportul conţine următoarele informaţii care ajută la înregistrarea datelor de identificare şi performanţă ale procesului de inspectie software:

- dispoziţia – se referă la procedura ce va fi folosită pentru verificarea refacerii produsului de către autor; valorile posibile sunt "*acceptare*", "*acceptare condiţionată*" şi "*reinspecţie*".
- informaţii despre produs – se înscriu următoarele informaţii:
  - numărul de inspectori;
  - numărul total de pagini inspectate;
  - numărul total de defecte găsite;
  - cine a realizat refacerea;
- informaţii despre efortul total depus – se descrie efortul în ore, pentru următoarele activităţi ale procesului de inspecţie:
  - efortul pentru planificare;
  - efortul pentru pregătire;
  - efortul total;
  - durata întâlnirilor de inspecţie;
  - efortul de completare a rapoartelor;
  - efortul de verificare;
- timpul de pregătire pentru inspectori – se prezintă timpul necesar fiecărui inspector pentru pregătire.
- certificarea moderatorului – este autorizarea dată de moderator precum că procesul de inspecţie a fost terminat.

34.12 Inspecţia software automată

În ultima vreme se afirmă tot mai mult tehnologiile de inspecţie software automată. Aceste tehnologii, livrate ca instrumente şi servicii comerciale, pot localiza multe erori de programare, erori care pot fi cauza celor mai multe eşecuri. Strategia acestor tehnologii este de a analiza codul sursă înainte de testarea sa şi de a identifica potenţialele probleme înainte ca acestea să se manifeste ca bug-uri de programare. Cel mai important aspect al inspecţiei automate este capacitatea de depanare a codului înainte de execuţia sa. Ea este superioară testării, care implică necesitatea executării codului şi a construirii, întreţinerii şi rulării de seturi de date de test.

Inspecţia software automată reprezintă o nouă abordare bazată pe utilizarea unor programe care realizează părţi din acest proces. Ea este mult mai eficientă decât inspecţia manuală şi asigură creşterea productivităţii în activitatea de dezvoltare de software. Acest lucru se realizează prin:

- reducerea costurilor, printr-o detectare mai puţin costisitoare a defectelor;
- reducerea timpului, printr-o detectare mai rapidă a defectelor;

- creşterea calităţii, prin depistarea defectelor ce nu pot fi identificate prin testare.

Unele instrumente folosite pentru inspecţia software automată, cum ar fi Flexelint de la Gimpel Software şi QAC de la Programming Research, verifică respectarea standardelor de codificare şi generează mesaje de atenţionare privind posibilele defecte. Unele instrumente generează un volum mare de mesaje de atenţionare care sunt fals pozitive. Cu alte cuvinte, instrumentul "crede" că a găsit un defect, dar la o analiză mai profundă a contextului se dovedeşte că acesta nu constituie o problemă reală. Toata lumea cunoaşte avertizările nerelevante generate de compilatoare. Această problemă fals pozitivă este destul de severă într-un instrument de inspecţie automată. În Flexelint de exemplu, apar peste 50 mesaje fals pozitive la fiecare defect real.

În anumite cazuri mesajele fals pozitive pot fi eliminate prin crearea de filtre capabile să înlăture automat un subset de astfel de mesaje. Totuşi este necesar un proces manual pentru a elimina mesajele fals pozitive nereţinute de filtru. Dezvoltatorii au nevoie de un mijloc de a evalua fiecare mesaj de atenţionare pentru a determina dacă este într-adevar un defect sau un mesaj fals pozitiv. Pentru a folosi efectiv instrumente de inspecţie automată, firmele dezvoltatoare de software trebuie să angajeze sau să pregătească experţi în inspecţie şi să implementeze o metodologie pentru evaluarea şi înlăturarea mesajelor fals pozitive, pentru a fi sigure că rezultatele inspecţiei automate conţin defecte reale şi nu un număr mare de mesaje fals pozitive.

Pentru asigurarea unui beneficiu maxim, inspecţia automată trebuie realizată imediat după terminarea etapei de codificare şi înainte ca produsul să intre în testare. Deoarece inspecţia automată nu necesită ca aplicaţia să fie compilată, programele pot fi inspectate chiar înainte de integrarea lor cu alte componente ale aplicaţiei.

Multe din eforturile de dezvoltare din zilele noastre sunt globale, cu echipe aflate în diferite locaţii lucrând în etapa de codificare. Cu inspecţia automată firmele de software pot asigura calitatea fiecărui segment de cod în fiecare etapă a procesului de dezvoltare. Fiecare componentă poate fi inspectată independent, apoi poate fi inspectată şi integrarea componentelor. Rezultatul va fi o soluţie "curată" de la început până la sfârşit.

O alternativă la utilizarea directă a unor instrumente de inspecţie automată este apelarea la serviciile unei firme specializate. Astfel, compania Reasoning Inc. pune la dispoziţie servicii prin care se identifică vulnerabilităţile şi defectele din programe realizate în limbajele C, C++ şi Java:

- zonele sensibile la accese neautorizate sau atacuri din afară;
- defecte care pot cauza căderi ale sistemului, coruperea datelor sau comportări anormale ale programelor;
- secvenţe de cod nefolosite sau incomplete, care pot avea impact negativ asupra integrităţii aplicaţiei şi a costurilor cu mentenanţa.

Realizarea unui produs software presupune un consum apreciabil de resurse umane şi financiare. Este necesar ca produsul să fie realizat în timpul planificat, cu nivelul de calitate stabilit prin cerinţele formulate şi în limita bugetului alocat [Ivan05]. Nici o echipă de dezvoltare software nu îşi poate permite ca produsul livrat să manifeste defecte şi/sau vulnerabilităţi

în faza de exploatare curentă. Înlăturarea lor în acest moment ar fi deosebit de costisitoare, atât în plan financiar cât şi în cel al imaginii.

Inspecţia software reprezintă cea mai eficientă tehnică utilizată în scopul asigurării unei înalte calităţi a produselor realizate. Este un proces de revizuire tehnică realizat de-a lungul etapelor de dezvoltare a produselor software, cu scopul de a identifica şi înlătura defectele. Ea se poate aplica oricărui produs rezultat în diversele etape ale procesului de dezvoltare: de la determinarea cerinţelor, până la etapa de testare. Scopul său este de a identifica şi înlătura defectele încă din primele etape ale procesului de dezvoltare, ştiind că remedierea lor în etapele ulterioare poate duce la creşterea costurilor de câteva zeci de ori.

Alături de testarea convenţională, inspecţia automată permite proiectelor software să obţină beneficiile combinării metodelor tradiţionale cu cele noi într-un mod echilibrat şi cu costuri reduse, astfel încât să se asigure o calitate superioară produselor software.