Fundamente de Inginerie Software

Cap. 7

Testare şi Validare.

Conf.Dr.Ing. Dan Pescaru

Textbooks: Sommerville "Software Engineering 7", 2004, Cap. 22, 23
Maciaszek "Practical Software Engineering", 2005, Cap. 12
Sursă: http://www.comp.lancs.ac.uk/computing/resources/IanS/SE7/
http://www.comp.mq.edu.au/books/pse/

2009

Verificare / Validare

- Verificare:

  "Construim (în mod) corect produsul?"
- Software-ul trebuie să fie conform cu specificaţiile

- Validare:

  "Construim produsul corect (care trebuie)?"
- Software-ul trebuie să facă ceea ce utilizatorul are nevoie

Procesul de Verificare şi Validare

- Procesul se desfăşoară pe întreaga durată de viaţă a sistemului – verificarea şi validarea trebuiesc aplicate fiecărei etape a procesului software

- Are două obiective principale:
  - Descoperirea defectelor din sistem
  - Evaluarea măsurii în care sistemul este uti şi utilizabil într-o situaţie operaţională

Scopul Verificării şi Validării

- Verificarea şi validarea trebuie să stabilească gradul în care software-ul se potriveşte scopului său
- Acest lucru nu înseamnă în mod necesar lipsa completă a defectelor
- Mai degrabă, el trebuie să fie suficient de bun pentru scopul în care a fost creat, iar felul utilizării va determina gradul de încredere necesar

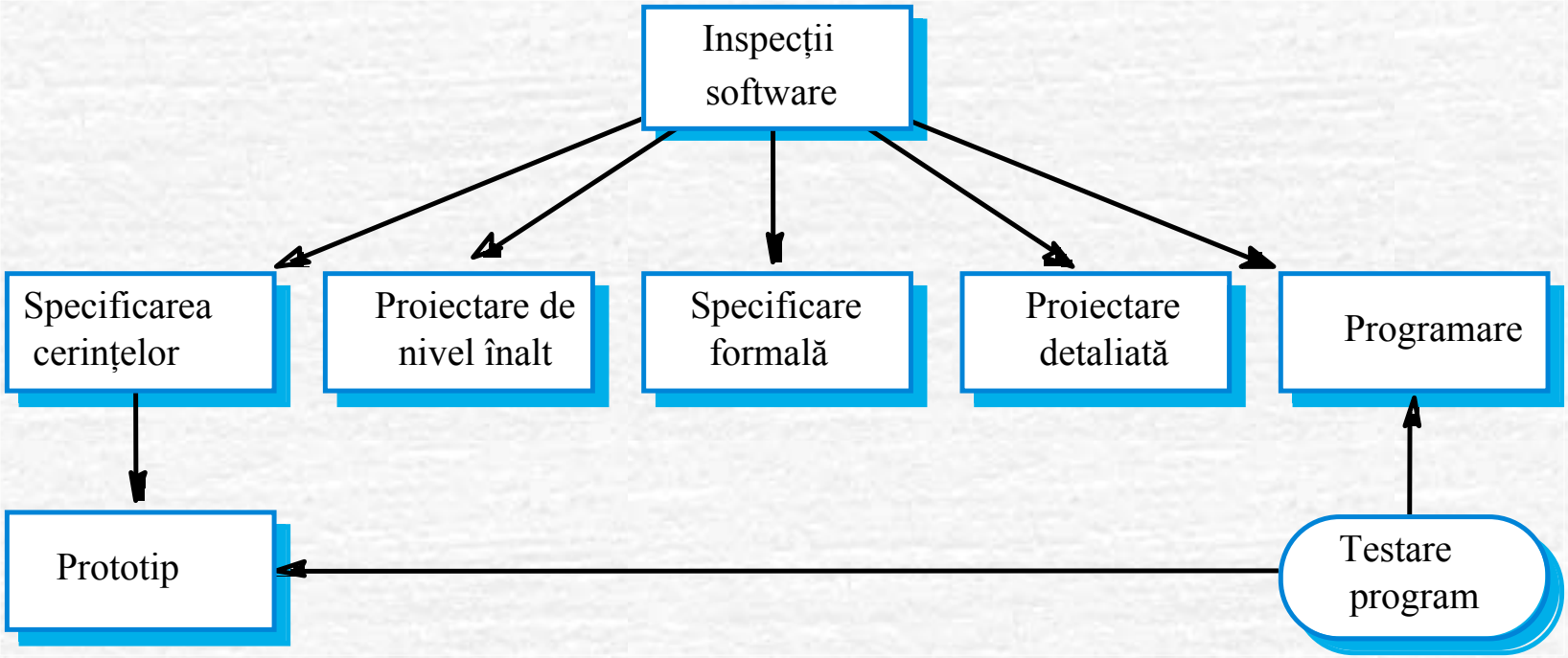
Gradul de încredere

- Depinde de scopul sistemului, aşteptările utilizatorilor şi condiţiilor de piaţă
  - Funcţia software-ului în companie
    - Nivelul încrederii depinde de cât de critic este sistemul pentru organizaţie
  - Aşteptările utilizatorilor
    - Utilizatorii pot avea aşteptări nu foarte ridicate de la anumite tipuri de software
  - Condiţiile de piaţă
    - Livrarea rapidă a unui produs software pe piaţă poate fi mai importantă decât găsirea tuturor defecţiunilor din program

Verificarea Statică şi Dinamică

- Inspectarea codului. Constă în analiza statică a reprezentării sistemului pentru descoperirea eventualelor probleme (verificare statică)
  - Poate şi suplimentată cu analiza automată a documentării şi codului realizată prin unelte dedicate
- Testarea software. Se preocupă de observarea comportamentului produsului la rulare (verificare dinamică)
  - Sistemul rulează cu date de test observându-se comportamentul operaţional

Verificarea şi Validare Statică şi Dinamică



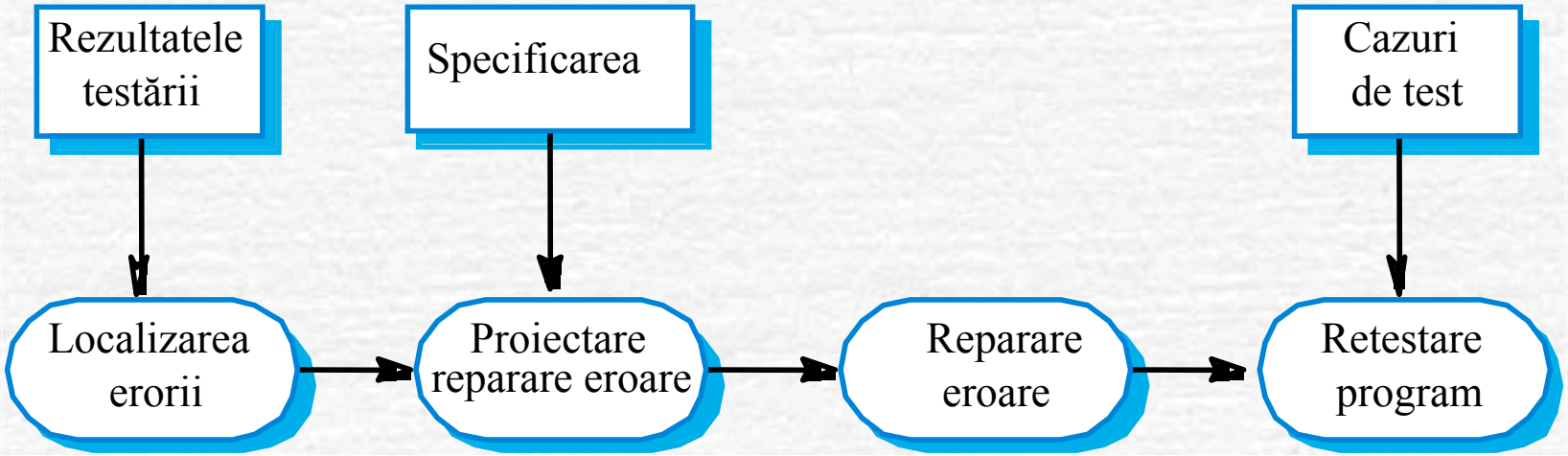
Testarea programului

- Poate scoate în evidenţă prezenţa unor erori (NU absenţa acestora!)
- Singura tehnică de validare pentru cerinţele non-funcţionale este executarea programului şi observarea comportamentului
- Trebuie utilizat în conjuncţie cu verificarea statică pentru completarea procesului de verificare şi validare

Testare şi Depanare

- Testarea de defecte şi depanarea sunt procese distincte
- Verificarea şi validarea se preocupă de stabilirea existenţei defectelor în program
- Depanarea se concentrează pe localizarea şi repararea erorilor
- Depanarea implică formularea unor ipoteze despre comportamentul programului care apoi sunt testate pentru a descoperi erori

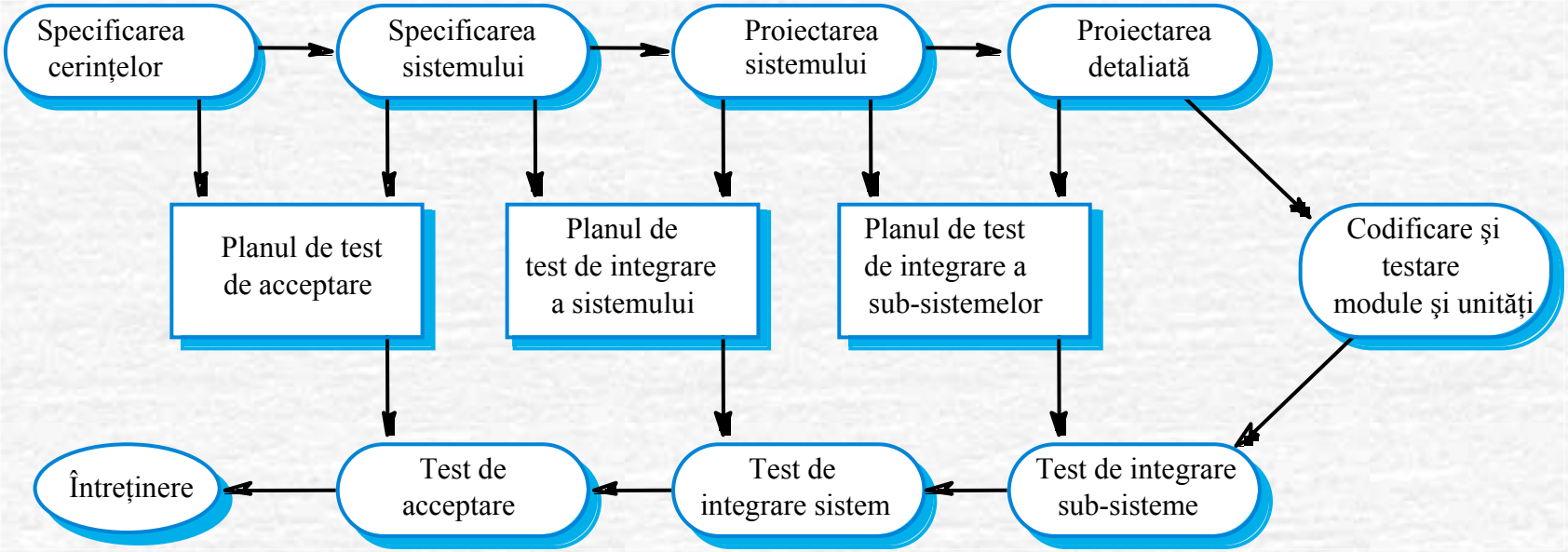
Procesul de Depanare



Planificarea Verificării

- Pentru a obţine rezultate optime este necesară o planificare atentă a testelor şi inspecţiilor
- Planificarea trebuie începută încă de la debutul procesului de dezvoltare
- Planul trebuie să identifice balansarea optimă între verificarea statică şi testare
- Prin planificare se urmăreşte definirea unor standarde pentru procesul de testare şi nu descrierea testărilor produsului

Modelul "V" de Dezvoltare



Structura unui Plan de Testare

- Procesul de testare. Fazele principale.
- Urmărirea cerinţelor. Fiecare cerinţă va fi testată individual
- Elementele testate. Specificarea produselor procesului software care vor fi testate
- Planificarea testării. Se precizează şi resursele alocate
- Procedurile de înregistrare a testării. Permit urmărirea corectitudinii efectuării testelor
- Cerinţe hardware şi software. Se precizează uneltele software utilizate pentru testare şi cerinţele lor
- Constrângeri. În special constrângeri legate de personalul insuficient disponibil pentru testare

Inspecţia codului

- Implică examinarea manuală a codului sursă în scopul descoperirii anomaliilor şi defectelor
- Inspecţiile nu necesită rularea sistemului aşa că pot fi făcute înainte de terminarea implementării
- Pot fi aplicate oricărei reprezentări a sistemului (cerinţe, proiect, date de configurare, date de test etc.)
- S-a dovedit ca fiind o tehnică eficientă de descoperire a erorilor în programe

Avantajele Inspecţiei

- Mai multe defecte pot fi descoperite la o singură inspecţie. La testare un defect poate masca altul aşa că sunt necesare rulări succesive

- Familiaritatea cu un domeniu şi cunoştinţele de programare ajută recenzorii să recunoască uşor anumite tipuri de erori care apar frecvent

Inspecţia şi Testarea

- Inspecţia şi testarea sunt tehnici de verificare complementare (şi nu opuse)
- Amândouă trebui utilizate pe timpul procesului de verificare şi validare
- Inspecţiile pot verifica conformitatea cu specificaţiile dar nu şi cu cerinţele reale ale utilizatorilor
- Inspecţiile nu pot verifica caracteristici non-funcţionale precum performanţa, uşurinţa la utilizare etc.

Procedura de Inspecţie

- Se începe cu o prezentare generală a sistemului echipei de inspecţie
- Codul şi documentaţia sunt distribuite echipei încă de la început
- Inspecţia are loc şi erorile descoperite sunt notate
- Se fac modificările care să repare erorile descoperite
- Repetarea inspecţiei poate fi câteodată necesară

Rolul Membrilor într-o Echipă de Inspecţie

- *Autorul*. Programatorul sau proiectantul responsabil cu codul sau documentul de inspectat. Va repara defectele descoperite la inspecţie
- *Inspectorul*. Găseşte erorile, omisiunile şi inconsistenţele în cod sau document. Poate identifica şi alte probleme în afara celor amintite
- *Cititorul*. Responsabil cu prezentarea codului la şedinţa de inspecţie
- *Secretarul*. Responsabil cu notarea rezultatelor şedinţelor de inspecţie
- *Moderatorul*. Gestionează şi facilitează inspecţiile. Raportează moderatorului şef
- *Moderatorul şef*. Responsabil cu îmbunătăţirea procesului, dezvoltarea standardelor etc.

Lista de Inspecţie

- Pentru conducerea inspecţiei se va utiliza o listă cu erori des întâlnite
- Lista de erori este dependentă de limbajul de programare şi reflectă erorile caracteristice acestuia
- In general cu cât verificările de tip sunt mai slab implementate de limbaj cu atât lista va fi mai lungă
- Exemple: iniţializări, nominalizarea constantelor, cicluri infinite, indici în afara marginilor tablourilor, etc

Ritmul Inspecţiei

- 500 de linii/oră în timpul privirii de ansamblu
- 125 linii de cod sursă/oră la pregătirea individuală
- 90-125 linii de cod/oră la inspectare
- Concluzie: inspecţia este un proces costisitor
- Inspecţia a 500 de linii de cod costa cam 40 oră-om

Automatizarea analizei statice

- Analizoarele statice sunt ustensile software care prelucrează surse text (cod sursă).
- Ele parcurs codul programului şi încearcă să descopere şi să raporteze potenţiale erori
- Sunt foarte eficiente în ajutorul inspecţiei – totuşi sunt doar un supliment şi nu pot înlocui inspecţia manuală

Exemple de teste la analiza statică

- Variabile utilizate înainte de iniţializare
- Variabile declarate dar neutilizate
- Variabile asignate de două ori dar cu prima valoare nefolosită
- Posibile depăşiri de margini la indicii de tablouri
- Variabile nedeclarate
- Cod la care execuţia nu poate ajunge
- Cicluri infinite
- Parametrii greşiţi la apelul de funcţii
- Pointeri neiniţializaţi

Etapele Analizei Statice

- Analiza fluxului de control. Verifică ciclurile cu intrări sau ieşiri multiple, caută cod la care execuţia nu poate ajunge etc.
- Analiza utilizării datelor. Detectează variabilele neiniţializate, variabilele declarate dar neutilizate etc.
- Analiza interfeţelor. Verifică consistenţa declarării şi apelului de funcţii şi proceduri

Etapele Analizei Statice

- Analiza fluxului informaţional. Identifică dependenţele variabilelor de ieşire. Nu detectează anomaliile în sine dar scoate în evidenţă informaţii pentru inspecţia de cod
- Analiza căilor. Identifică căile de rulare prin program şi delimitează instrucţiunile executate pe acea cale. Din nou este utilă în procesul de inspecţie
- Ambele etape generează o mare cantitate de informaţii. Ca atare trebuie utilizate cu grijă

Utilitatea Analizei Statice

- Foarte valoroasă când limbajul utilizat este mai slab tipizat (ex. C) şi ca atare multe erori trec nedetectate de compilator

- Mai puţin eficientă pentru limbaje precum Java care au o testare puternică a tipurilor şi ca atare detectează multe erori în timpul compilării

Procesul de Testare

- Testarea Componentelor
  - Testarea individuală a fiecărei componente a programului
  - În mod obişnuit este responsabilitatea dezvoltatorului de componente (cu unele excepţii la sistemele critice)
  - Testele sunt derivate din experienţa dezvoltatorului
- Testarea Sistemului
  - Testarea grupurilor de componente integrate într-un sistem sau sub-sistem
  - Responsabilitatea unei echipe de testare independente
  - Testele sunt bazate pe specificarea sistemului

Fazele Testării

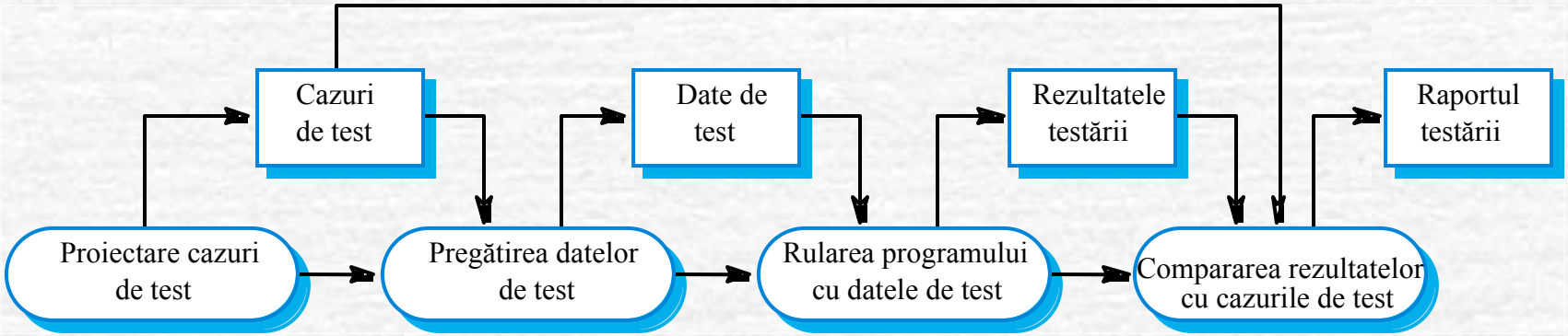
Dezvoltatorul software

Echipă de testare independentă

Tipuri de Testări

- Testarea de defecte
  - Teste proiectate să descopere defectele sistemului
  - Un test de defecte are succes dacă relevă prezenţa acestora în sistem
  - Testele relevă prezenţa defectelor şi NU absenţa lor
- Testarea de validare
  - Are ca scop să arate că sistemul se potriveşte cerinţelor
  - Testul se consideră că are succes dacă arată că cerinţele au fost implementate corespunzător

Procesul de Testare Software



Politici de Testare

- Pentru a demonstra lipsa defectelor dintr-un program ar trebui făcută o testate exhaustivă. Cu toate acestea testarea exhaustivă este practic imposibilă.

- Politicile de testare definesc abordarea care va fi utilizată la selectarea testelor de sistem:
  - Toate funcţiile accesibile din meniu trebuie testate
  - Combinaţiile de funcţii accesibile din acelaşi meniu trebuie testate
  - Toate funcţiile care necesită intrări de la utilizator trebuie testate atât cu date de intrare corecte cât şi incorecte

Testarea Sistemului

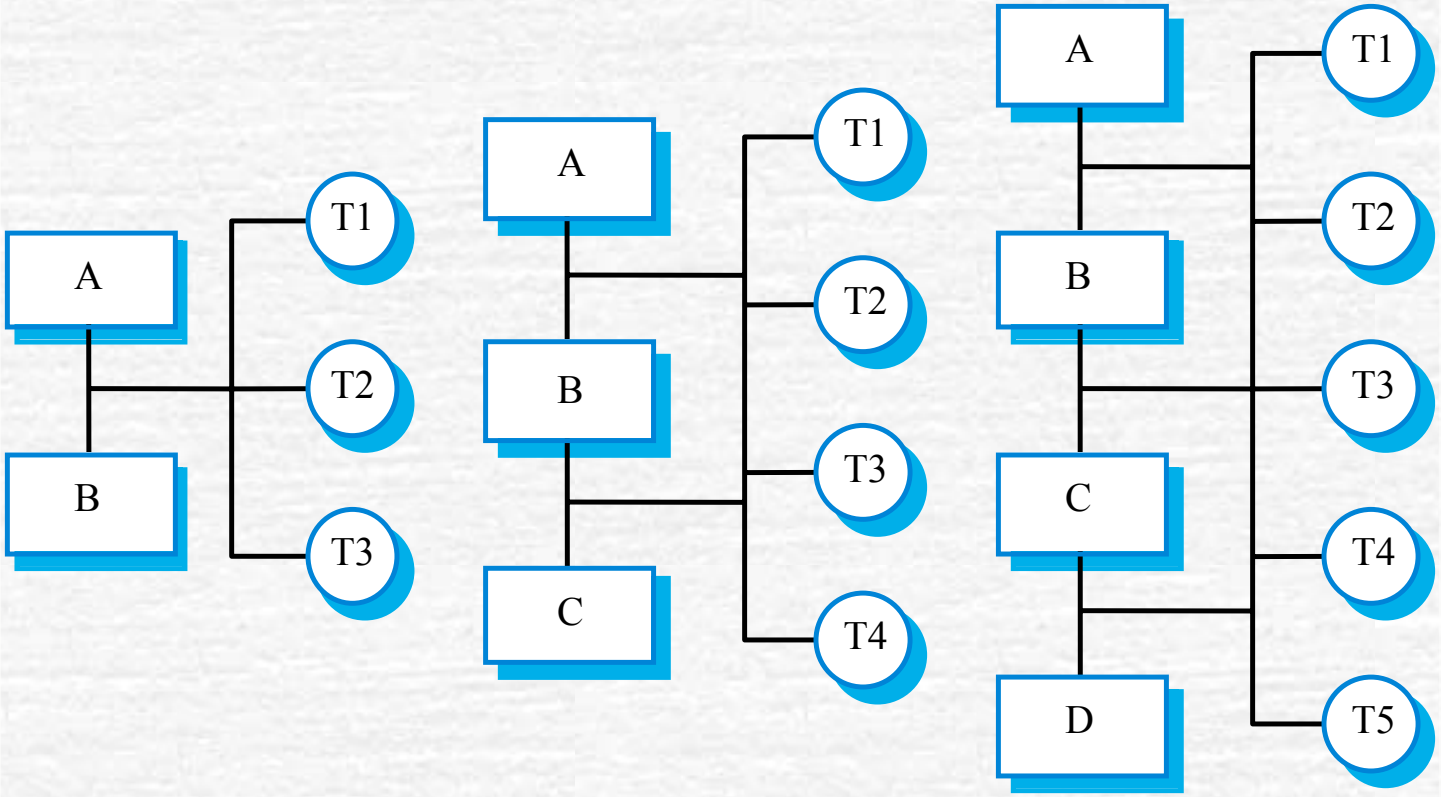
- Implică integrarea componentelor pentru a crea sub-sisteme sau sistemul respectiv

- Poate implica testarea unei variante care să fie livrate la client
- Două faze:
  - Testarea de integrare – echipa de testare are acces la codul sursă al sistemului. Sistemul este testat la nivel de componente pe măsură ce acestea sunt integrate
  - Testarea variantei finale – echipa de testare testează sistemul complet livrat ca un black-box

Testarea de Integrare

- Implică crearea sistemului din componentele sale şi testarea lui pentru a detecta problemele care apar din interacţiunea componentelor
- Integrarea top-down
  - Se dezvoltă un schelet al sistemului pe care se plasează componentele
- Integrarea bottom-up
  - Se integrează mai întâi componentele de infrastructură şi apoi se adaugă componentele funcţionale
- Pentru a simplifica localizarea erorii, sistemul trebuie integrat incremental

Testarea cu Integrare Incrementală

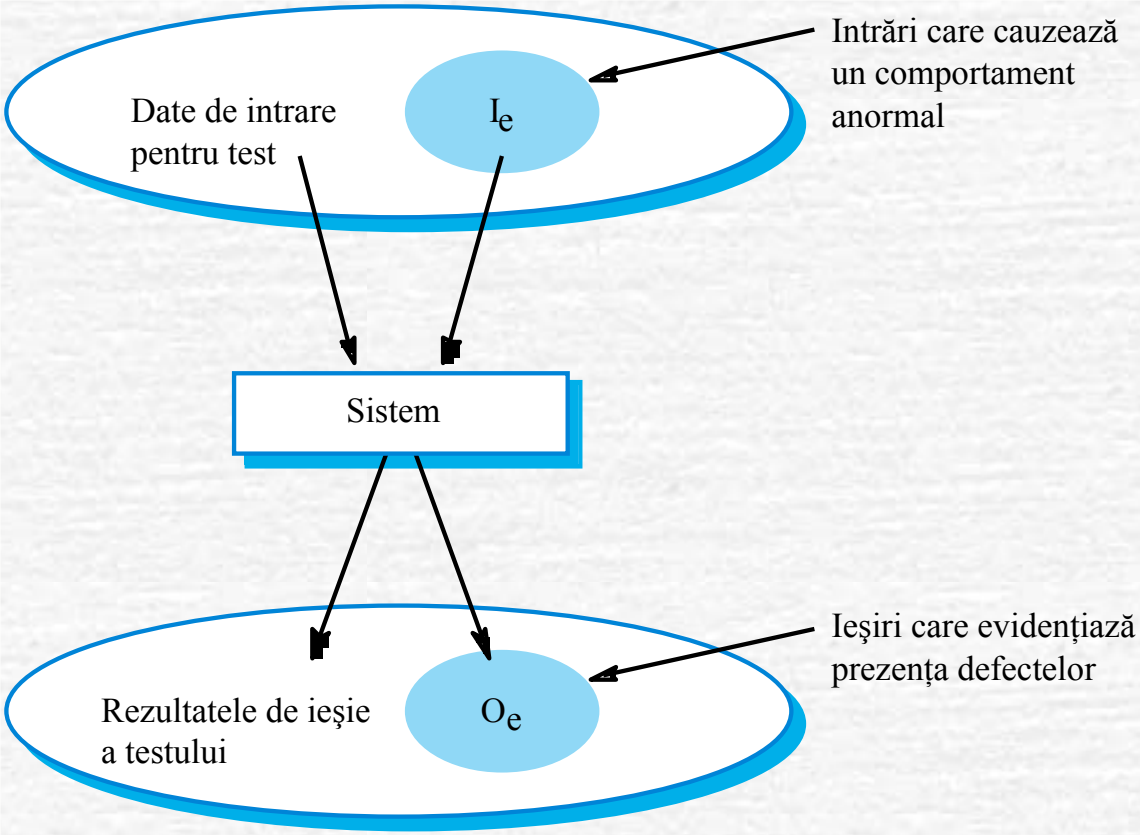


Secvența de Test 1 Secvența de Test 2 Secvența de Test 3

Abordări

- Validarea arhitecturii
  - Testarea cu integrarea top-down este utilă la descoperirea erorilor în arhitectura sistemului
- Demonstrarea sistemului
  - Testarea cu integrarea top-down permite demonstrarea limitată a funcţionării sistemului încă din fazele de început
- Testarea implementării
  - Deseori este mai simplă la testarea cu integrare bottom-up
- Observarea testelor
  - Apar probleme în cazul ambelor tehnici. În general este nevoie de cod extern pentru a observa testele

Testarea Black-Box



Sfaturi Utile la Testare

- Aceste sfaturi sunt utile pentru echipa de testare în ai ajuta să aleagă testele care pot evidenţia defecte în sistem
  - Alegerea unei intrări care forţează sistemul să genereze toate mesajele de eroare
  - Alegerea intrărilor care cauzează depăşirea memoriei bufferelor
  - Repetarea aceleiaşi intrări sau a unei serii de intrări succesiv de câteva ori
  - Forţarea ieşirilor invalide
  - Forţarea rezultatelor calculelor să fie prea mici sau prea mari

Teste de performanţă

- O parte a testării versiunilor ce vor fi distribuite clienţilor implică testarea unor proprietăţi emergente precum performanţa sau fiabilitatea
- Testele de performanţă implică în mod uzual o serie de teste la care încărcarea este mărită constant până când performanţa sistemului devine inacceptabilă

Teste de Stres

- Forţează sistemul deasupra încărcării maxime prevăzute la proiectare. Condiţiile de stres scot deseori la lumină defecte ale sistemului
- În cazul unor condiţii de stres sistemul ar trebui să nu dea greş în mod catastrofal. În acest fel se pot testa pierderile inacceptabile de date sau servicii
- Aceste teste sunt în deosebi relevante la sistemele distribuite care pot prezenta degradări severe dacă reţeaua devine supraîncărcată

Testarea Componentelor

- Procesul de testare a componentelor presupune teste individuale executa asupra componentelor izolate
- Este un proces de testare la defecte
- Componentele pot fi:
  - Funcţii sau metode individuale ale unui obiect
  - Clase de obiecte având câteva atribute ţi metode
  - Componente compuse împreună cu interfeţele utilizate pentru accesarea funcţionalităţilor lor

Testarea Obiectelor

- Testele complete pentru o clasă implică
  - Testarea tuturor operaţiilor asociate cu obiectele sale
  - Setarea şi citirea tuturor atributelor unui obiect
  - Încercarea obiectului în toate stările sale posibile
- Moştenirea face mult mai dificilă proiectarea unor teste pentru clase şi obiecte deoarece informaţia de testat nu este localizată doar la clasa respectivă

Testarea Interfeţelor

- Obiectivele sunt de a detecta erori datorate interfeţelor sau presupunerilor greşite despre interfeţe
- Acestea sunt importanta mai ales pentru dezvoltarea orientată pe obiecte deoarece obiectele sunt definite de interfeţele lor

Tipuri de Interfeţe

- Iterfeţe de parametrii
  - Datele trimise de la o procedură la alta
- Interfeţe cu memorie comună
  - Un bloc de memorie este folosit în comun de proceduri sau funcţii
- Interfeţe procedurale
  - Sub-sistemul înglobează un set de proceduri pentru a fi apelat de late sub-sisteme
- Interfeţe cu schimb de mesaje
  - Sub-sitemele solicită servicii de la alte sub-sisteme

Erori ale Interfeţelor

- Utilizare greşită a interfeţei
  - Apare când o componentă cheamă altă componentă şi comite o eroare în utilizarea interfeţei acesteia, de ex. trimite parametrii într-o ordine greşită
- Neînţelegerea interfeţei
  - O componetă înglobează nişte presupuneri incorecte despre comportamentul componentei chemate
- Erori de sincronizare
  - Componetele care interacţionează operează la viteze diferite şi ca atare informaţia accesată poate fi expirată

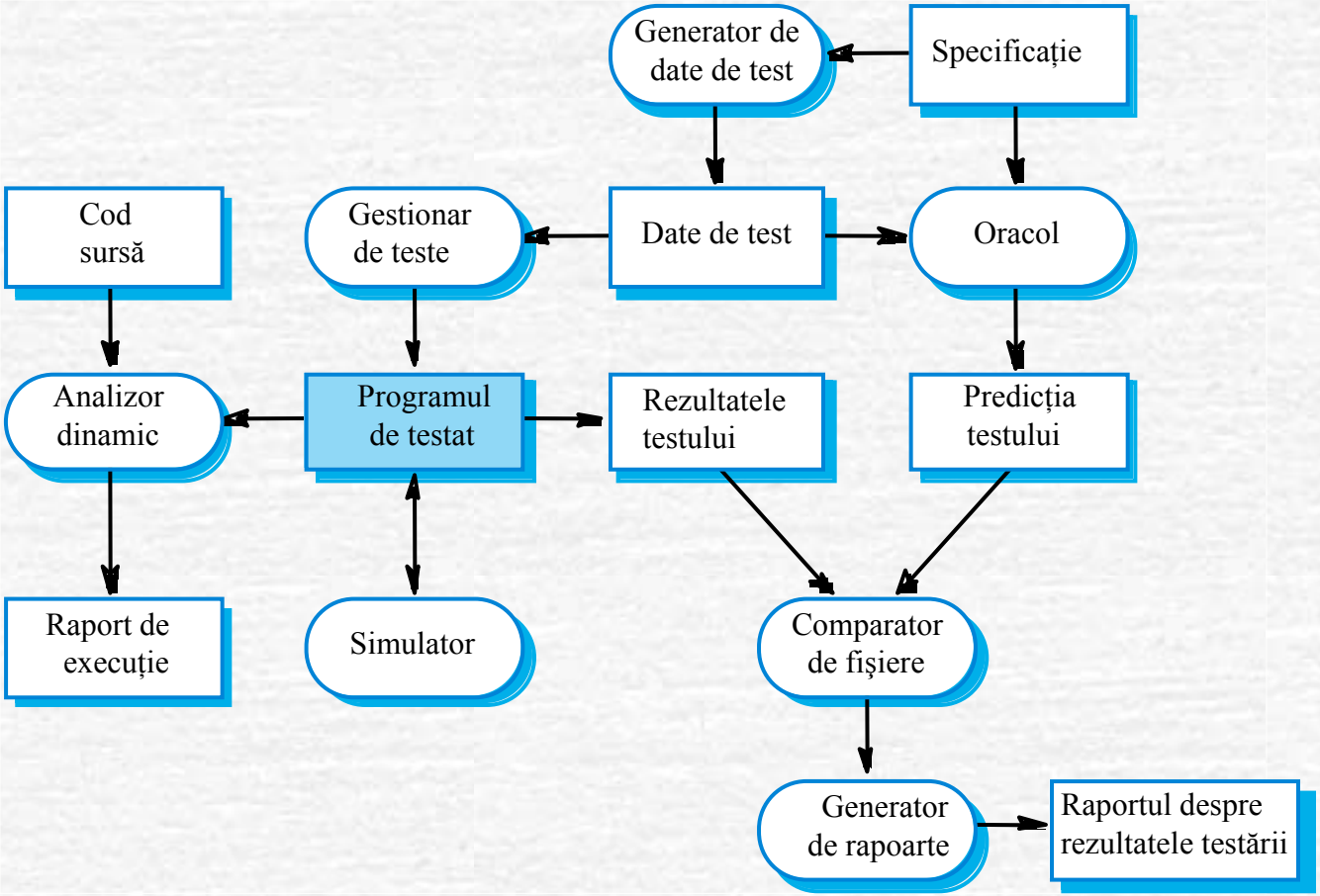
Sfaturi la Testarea Interfeţelor

- Testele vor fi proiectate astfel încât parametrii de apel ai procedurilor să fie aleşi la marginea intervalelor lor
- Parametrii pointer vor fi testaţi cu valori NULL
- Testele vor fi astfel proiectate ca să forţeze componenta să dea greş
- Se vor utiliza teste de stres în mecanismul de schimb de mesaje
- La sistemele cu memorie comună se va varia ordinea în care componentele sunt activate

Automatizarea Testării

- Testarea este o fază a procesului costisitoare. Sistemele de testare oferă o gamă variată de ustensile care reduc timpul necesar şi ca atare costul total al testării
- Sisteme precum Junit sprijină execuţia automată de teste
- Majoritatea sistemelor de testare sunt sisteme deschise deoarece cerinţele referitoare la testare sunt specifice fiecărei organizaţii
- Câteodată este dificil de integrat sistemele de testare la proiectarea aplicaţiei

Structura unui Sistem de Testare



Concluzii

- Verificarea şi validarea nu sunt unul şi acelaşi lucru. Verificarea arată conformitatea cu o specificaţie pe când validarea arată acoperirea nevoilor clienţilor
- Pentru a ghida procesul de testare este nevoie de un plan de testare
- Tehnicile de verificare statică implică examinarea şi analiza codului programului pentru detectarea erorilor
- Testarea poate arăta prezenţa erorilor în sistem dat nu poate demonstra lipsa erorilor

Concluzii

- Dezvoltatorii componentelor sunt responsabili de testarea acestora. Testarea sistemului este responsabilitatea unei echipe separate
- Testarea de integrare se ocupă cu testarea variantelor incrementale ale sistemului
- Testarea de defecte se proiectează pornind de la experienţa echipei şi a unor linii de ghidaj generale
- Testarea interfeţelor are ca scop descoperirea defectelor interfeţelor componentelor compuse