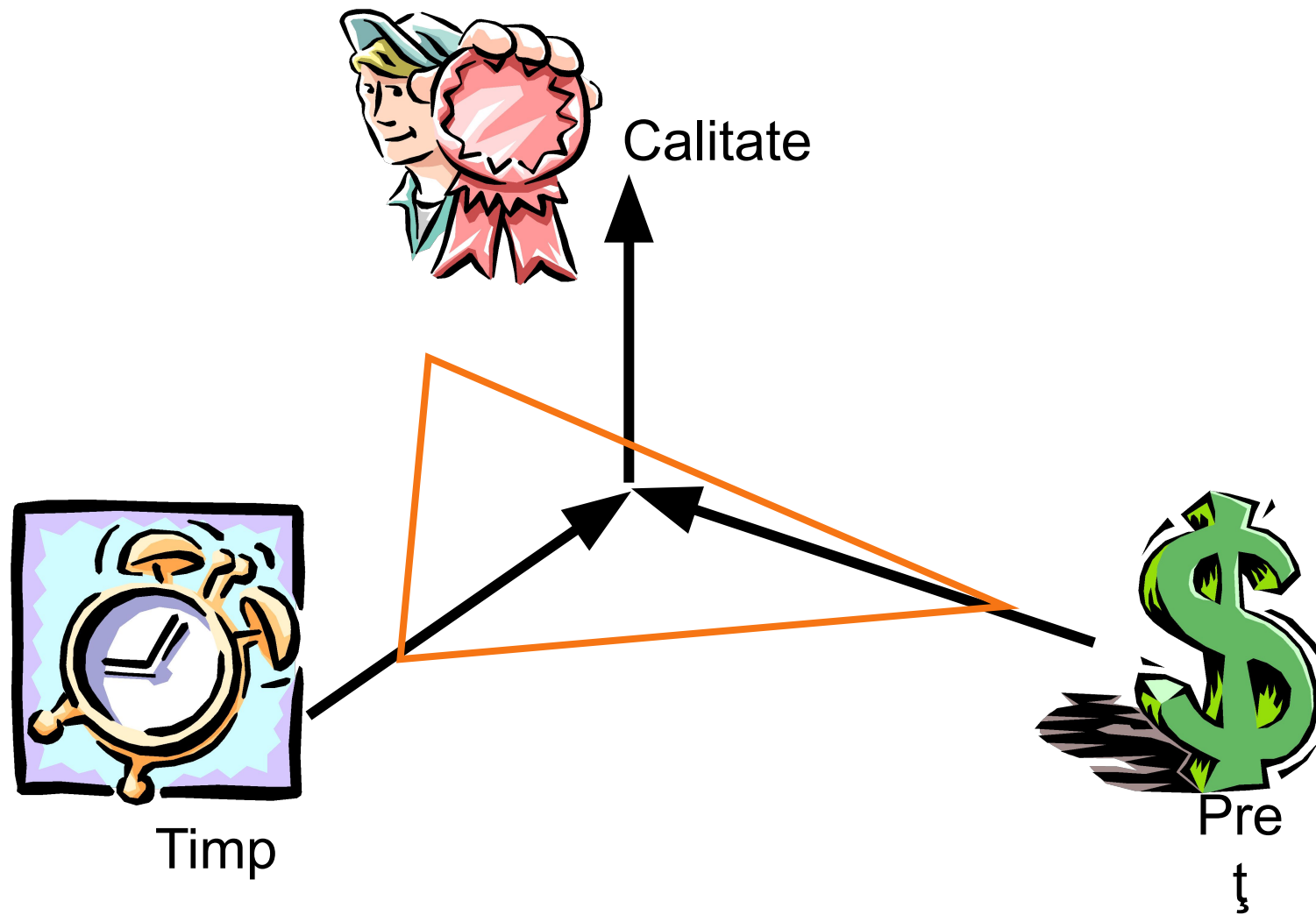


Introducere în Testare

Dilema Calității Software



Testare Software - Definiție

“The process of exercising or evaluating a system by manual or automated means to verify that it satisfies specified requirements or to identify differences between expected and actual results.”

(IEEE Standard Glossary, 1983)

Testare Software

- Testarea Software **NU** este o fază
- Este un proces care trebuie integrat în toate fazele construcției produsului software
- Există documente de testare asociate la fiecare fază a dezvoltării

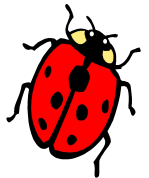
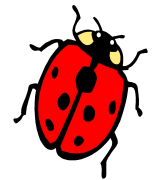
Care sunt Scopurile Testării?

- De a localiza și preveni “*bugs*” cât mai curând posibil
- De a efectua toate testele corespunzător cerințelor, într-un mod cât mai eficient și mai economic
- De a aduce produsul software la un nivel de calitate cât mai ridicat (pentru client)

**Toate acestea se execută folosind
*Metodologiile de Implementare***

De ce avem “bugs” în Software?

- Comunicarea deficitară sau blocajele de comunicare
- Înțelegerea deficitară
- Presiunea timpului
- Nivelul programatorului este scăzut

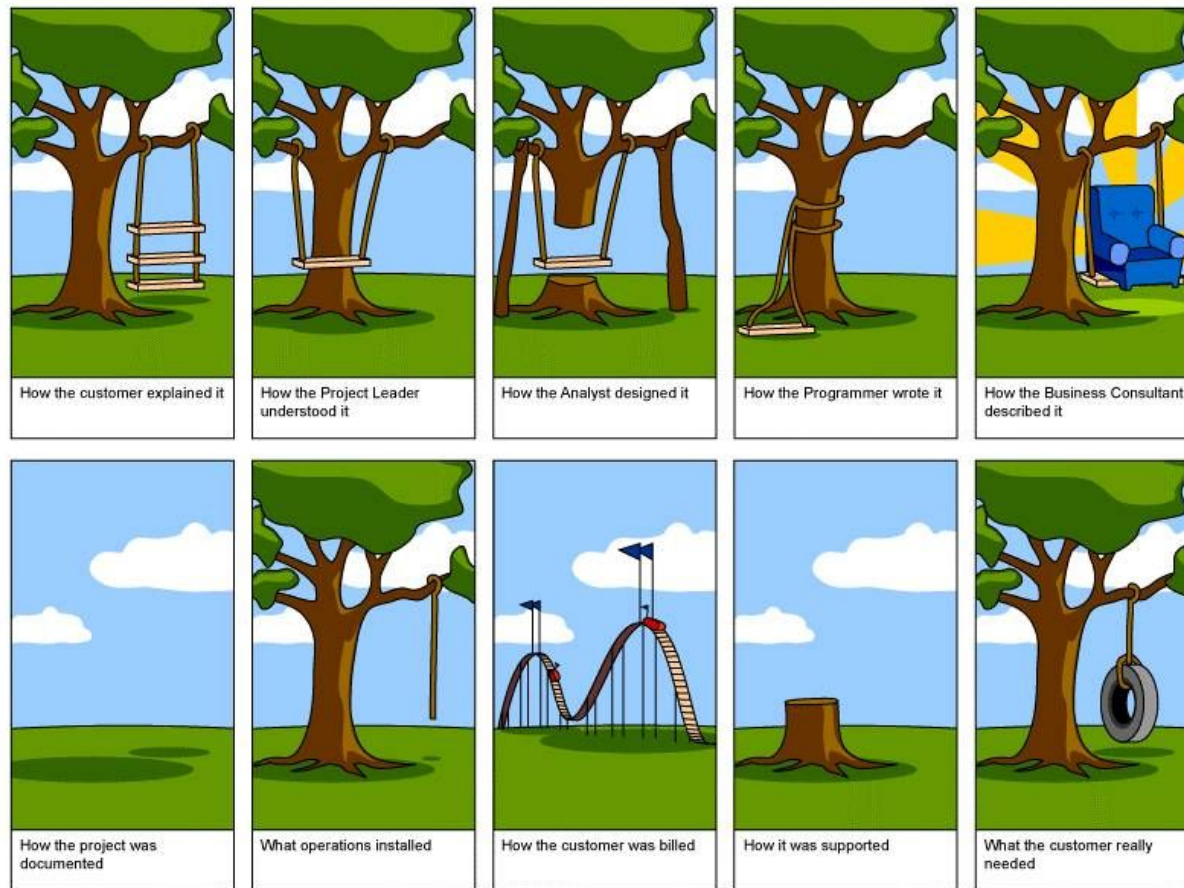


Comunicare Deficitară

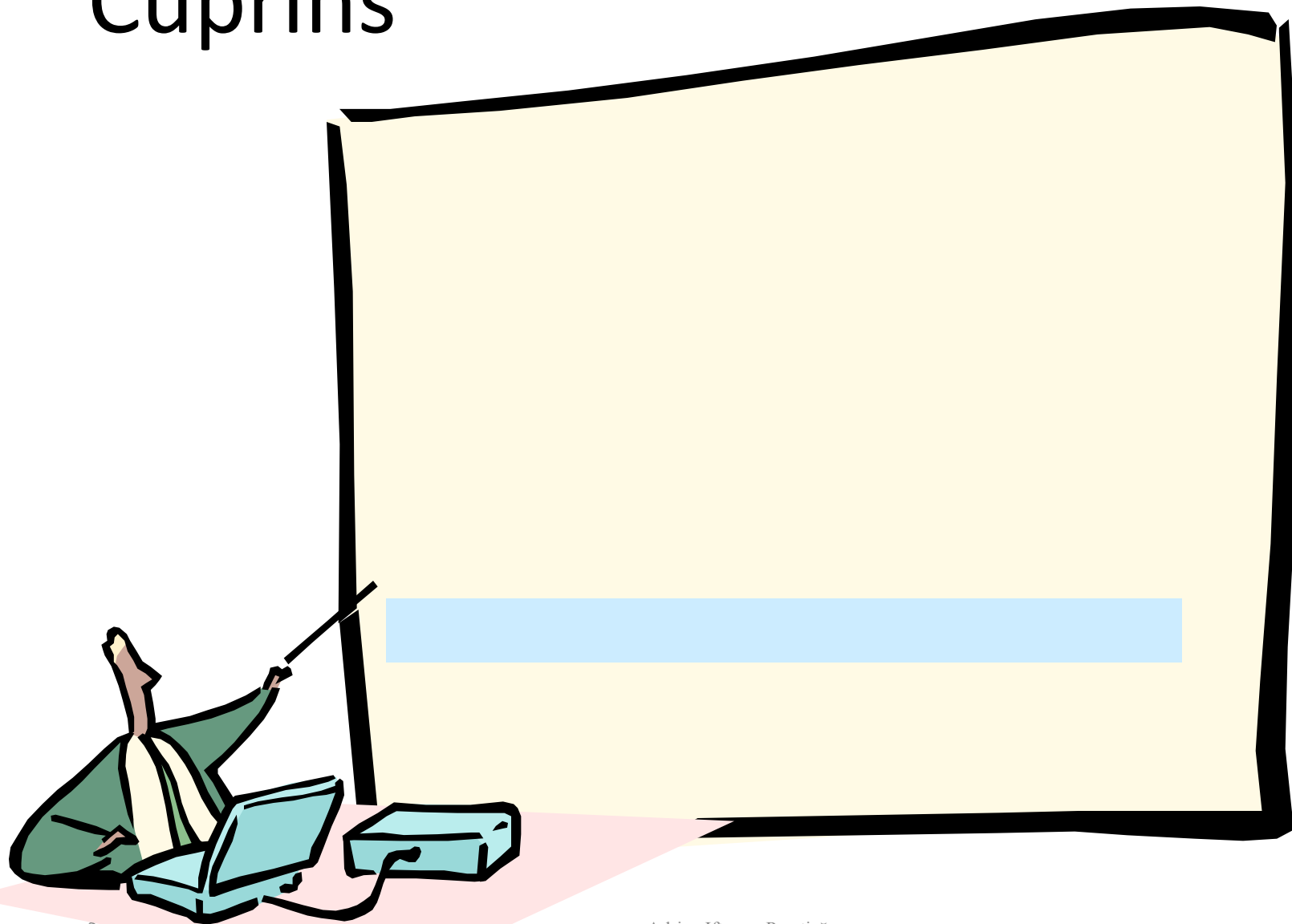


"Didn't you get my e-mail?"

Comunicare deficitară – În tratarea cerințelor



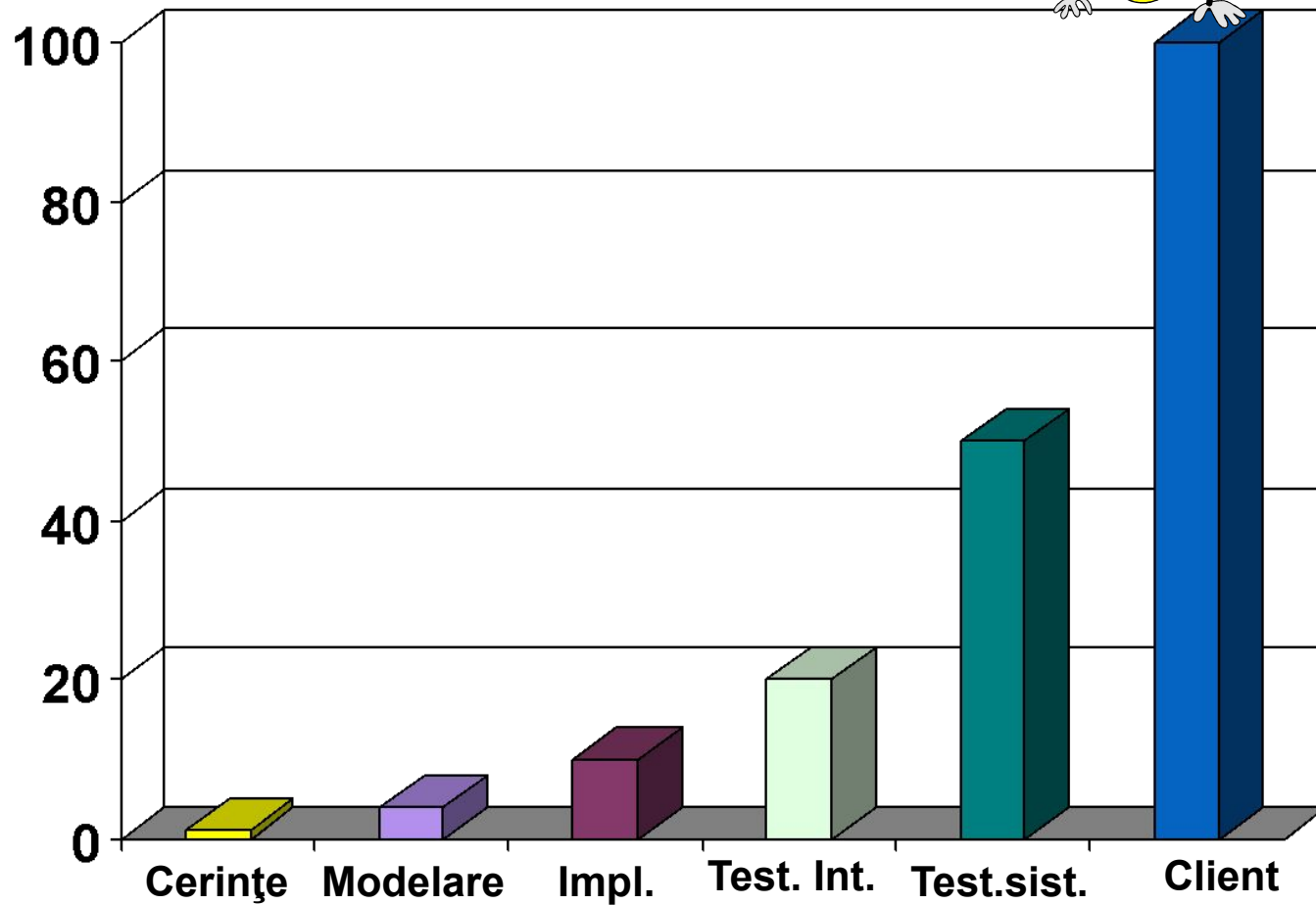
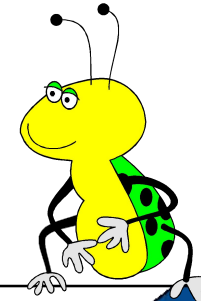
Cuprins



De unde vin Problemele Software?

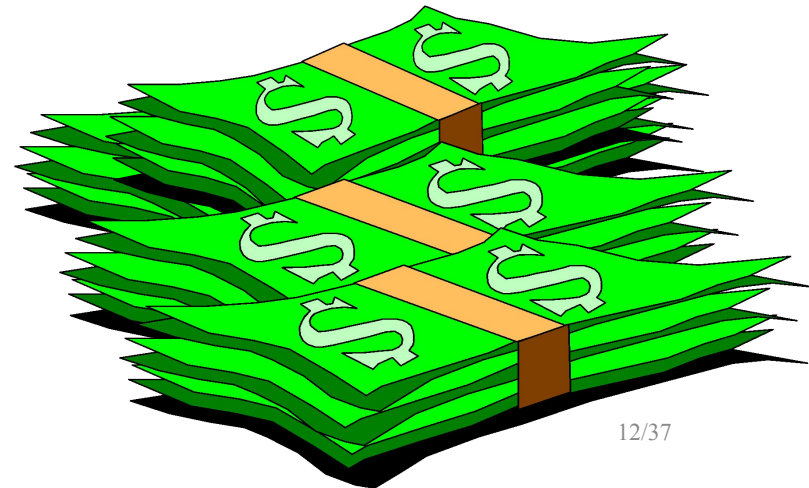
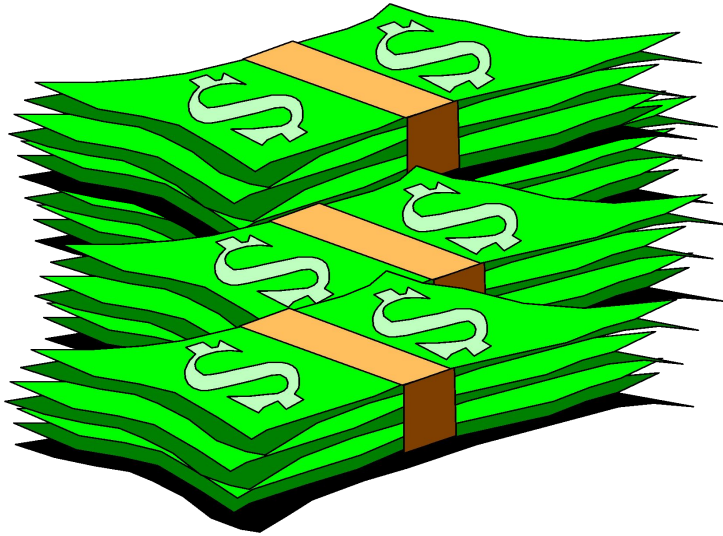
- Cerințe definite incomplet 50%
- Modelare ambiguă sau insuficientă 30%
- Erori de programare 20%

Bugs - Costul Fixării



Atenție

Găsirea târzie a bugs \Rightarrow un cost cât mai mare pentru a le fixa



Erori? Trebuie fixate cât mai Devreme Posibil



CERINȚE MODELARE IMPLIM. TESTARE CLIENT

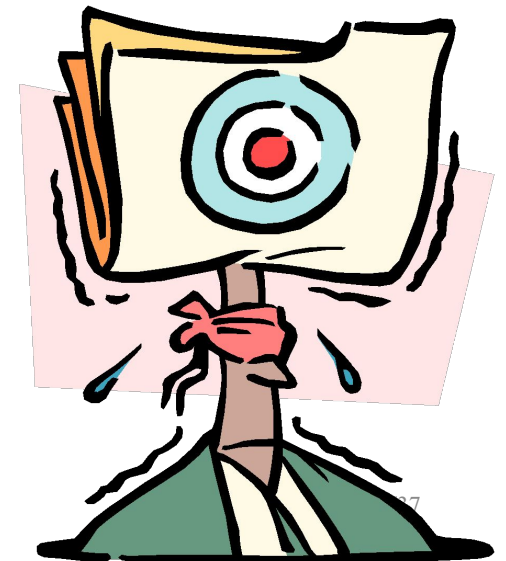
Testare Profesională

Profesionalismul în testare constă în abilitatea de a selecta numărul minim de cazuri de testare eficiente ce va fi capabil să verifice numărul maxim de funcții ale sistemului.

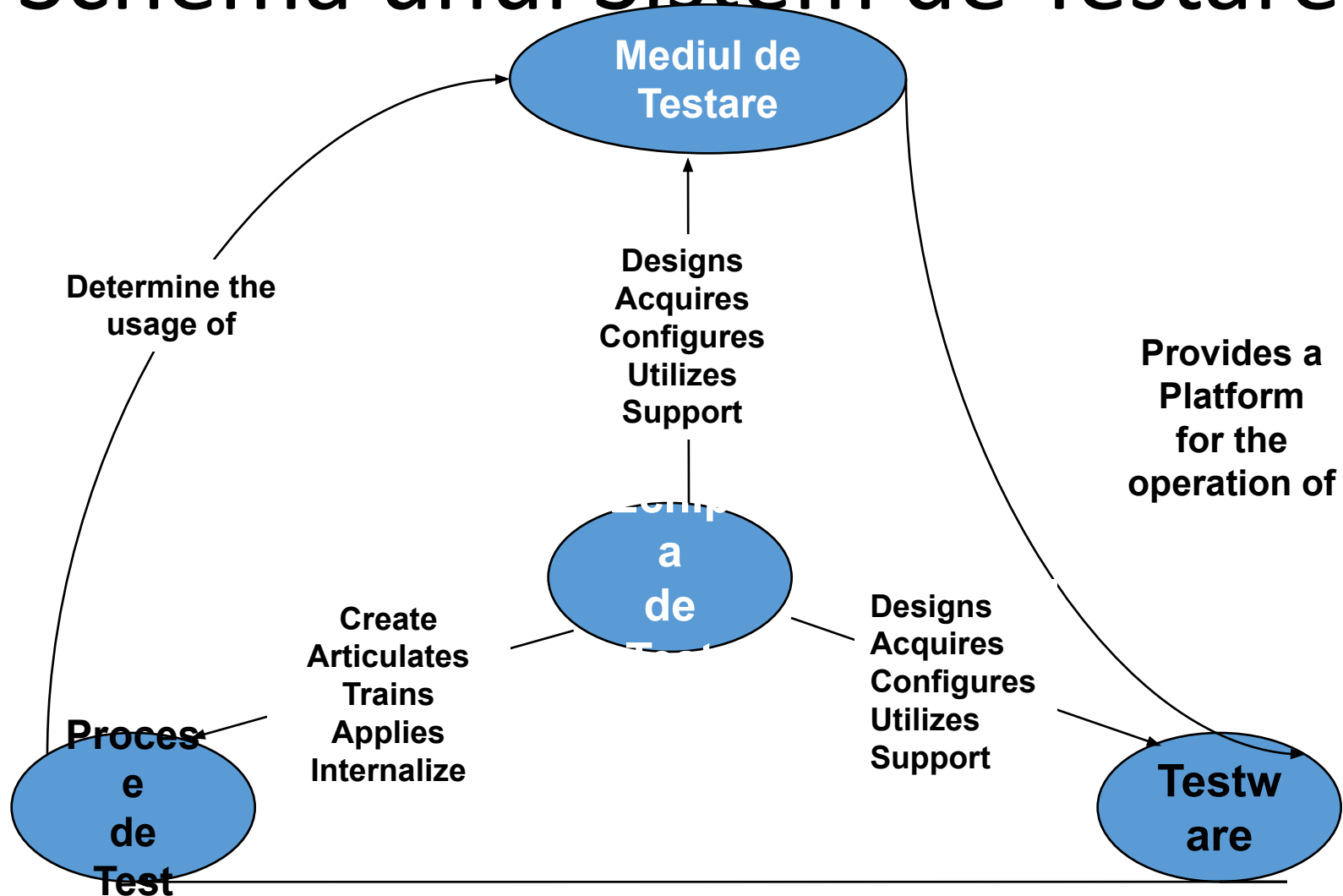


Când Oprim Testarea?

- Niciodată
- Când numărul de erori găsite într-un ciclu de testare este mai mic decât un număr stabilit
- Când nu mai sunt găsite defecte critice și majore
- Când timpul a expirat



Schema unui Sistem de Testare



Metodologii de Testare

Conținut

- Diferența dintre testare SW și debug SW
- Nivele de Test
- Clase de Test
- Conținutul Testării
- Testare și Dezvoltare SW

Diferența dintre testare software & debug

Testare

- Verificarea respectării cerințelor
- De regulă e făcută de o entitate externă și neutră
- Este un proces planificat și controlat

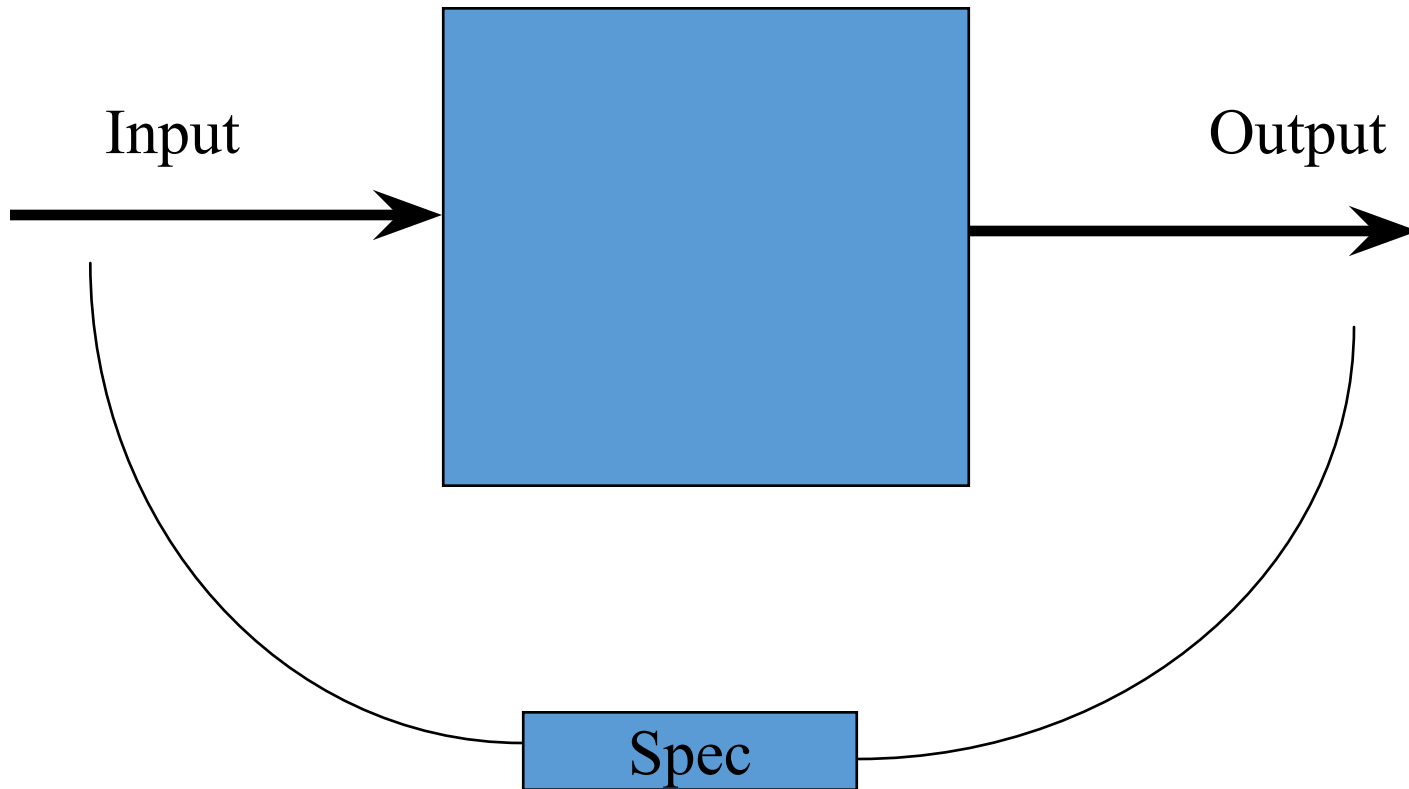
Debug

- Verificarea validității secțiunilor
- E făcută de programator
- E un proces aleator

Nivele de Test

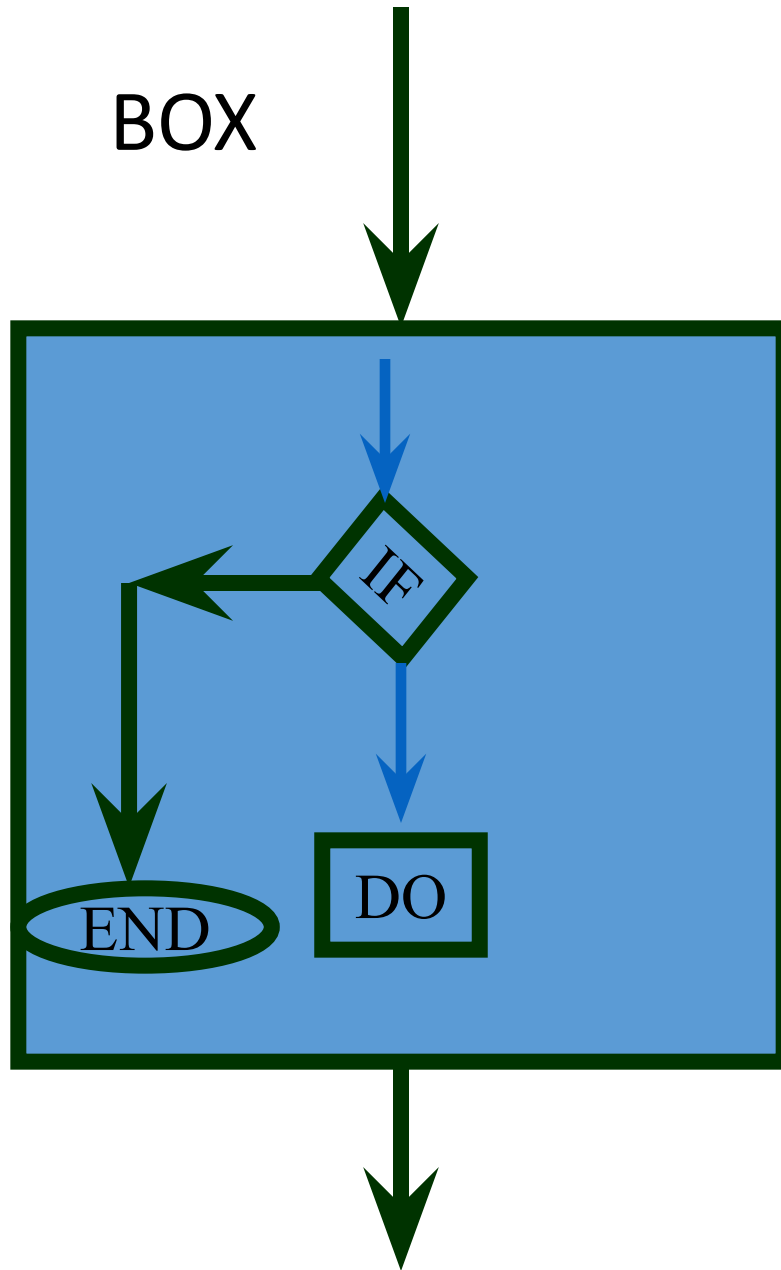
- Unitate sau Debug.
- Modul/Sub-Sistem.
- Integrare.
- Sistem.
- Acceptare.

BLACK BOX



WHITE

BOX



Unit Testing

- **Testarea unei funcții, a unui program, a unui ecran, a unei funcționalități**
- **Se face de către programatori**
- **Predefinită.**
- **Rezultatele trebuie documentate**
- **Se folosesc simulatoare pentru Input și Output**

Testare la Integrare

- **Testarea funcționării unor module în același timp**
- **Testarea coexistenței**
- **Se execută de către programatori sau de către testări analiști**
- **Testare pre-planificată**
- **Rezultatele se documentează**

Testare Automată vs Testare Manuală

- *Se găsesc rapid problemele*
 - *Se câștigă timp când e nevoie să repetăm testele*
 - *Procesul de scriere a codului e mult mai flexibil*
 - *Reduce volumul de testare manuală*
 - *Dezvoltarea software devine previzibilă și repetabilă*
- *Rezolvă problemele de interfață: scrierea corectă a textelor, mesajelor, aranjarea corectă în pagină, în ordinea care trebuie, sunt vizibile, etc.*
 - *Realizarea Scenariilor de test poate fi o treabă de durată și anevoioasă și implică o cunoaștere temeinică a întregului sistem*

Links

- <http://www.automatedqa.com/techpapers/testing.asp>
- <http://www.codeproject.com/tools/tilo.asp>
- <http://www.parasoft.com/jsp/products/home.jsp?product=Cpp>
- http://www.verifysoft.com/en_ctapp.html
- <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dncdev00/html/vc00f6.asp>
- <http://www.codeproject.com/gen/design/autp5.asp>
- <http://www.codeproject.com/cpp/UnitTestsReporter.asp>
- <http://www.codeproject.com/gen/design/onunittesting.asp>
- <http://www.code-agazine.com/Article.aspx?quickid=041103>

Coding Style – Motivație

- **Convențiile de programare sunt importante deoarece:**
- **80% din timpul alocat unei componente software este întreținere**
- **Foarte rar un produs software este întreținut pe toată durata folosirii lui de către aceeași persoană**
- **Convențiile de cod îmbunătățesc lizibilitatea produsului, și permite inginerilor software să înțeleagă rapid un program nou**

Coding Style - Cerințe

- **Folosirea fără rezerve a Comentariilor: ce fac procedurile, ce reprezintă variabilele, explicarea pașilor algoritmului, etc.**
- **Folosirea numelor sugestive pentru variabile si proceduri**
- **Scrierea modulara a proiectului**
- **Folosirea perechilor de tip set/get, start/stop, adauga/sterge, salvare/incarcare**

Coding Style - Links

- **C++:**

- <http://www.chris-lott.org/resources/cstyle/>
- <http://geosoft.no/development/cppstyle.html>

- **Java:**

- <http://java.sun.com/docs/codeconv/>
- <http://geosoft.no/development/javastyle.html>

| | | |
|----------------------------------------------------|------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Complexitatea proiectului | A: 2p B: 1p C: 0p | Un proiect care necesită o cantitate mai mare de muncă va fi recompensat mai bine. Se ia în calcul și originalitatea ideii precum și alegerea corespunzătoare a tehnologiei de implementare. Acolo unde este cazul se ia în considerare munca necesară implementării interfeței și calitatea rezultatului. |
| Fișa cerințelor, raportul final, activitate | A: 2p B: 1p C: 0p | Se punctează modul în care a fost realizată fișa cerințelor și felul în care studentul a interacționat cu conducătorul de laborator ("clientul"). Un punctaj bun se acordă dacă clientului îi este clar cu ce se ocupă proiectul și i-a fost prezentată evoluția proiectului. |
| Stil de programare | A: 1p B: 0.5p C: 0p | Se punctează aderarea la stilul de programare menționat în fișierul raport.html, denumirea auto-descriptivă a variabilelor etc. În această categorie intra și folosirea adecvată a unui sistem de generare automată a documentației (javadoc, doxygen etc.) |
| Proiectare și modularitate | A: 1p B: 0.5p C: 0p | Se punctează modul în care este structurat proiectul, posibilitatea de a reutiliza cod în alte proiecte. |
| Scenarii de Test | A: 1p B: 0.5 C: 0p | Se punctează folosirea unităților de testare automată în cadrul proiectului (JUnit, cppunit, NUnit, phpunit ...) |