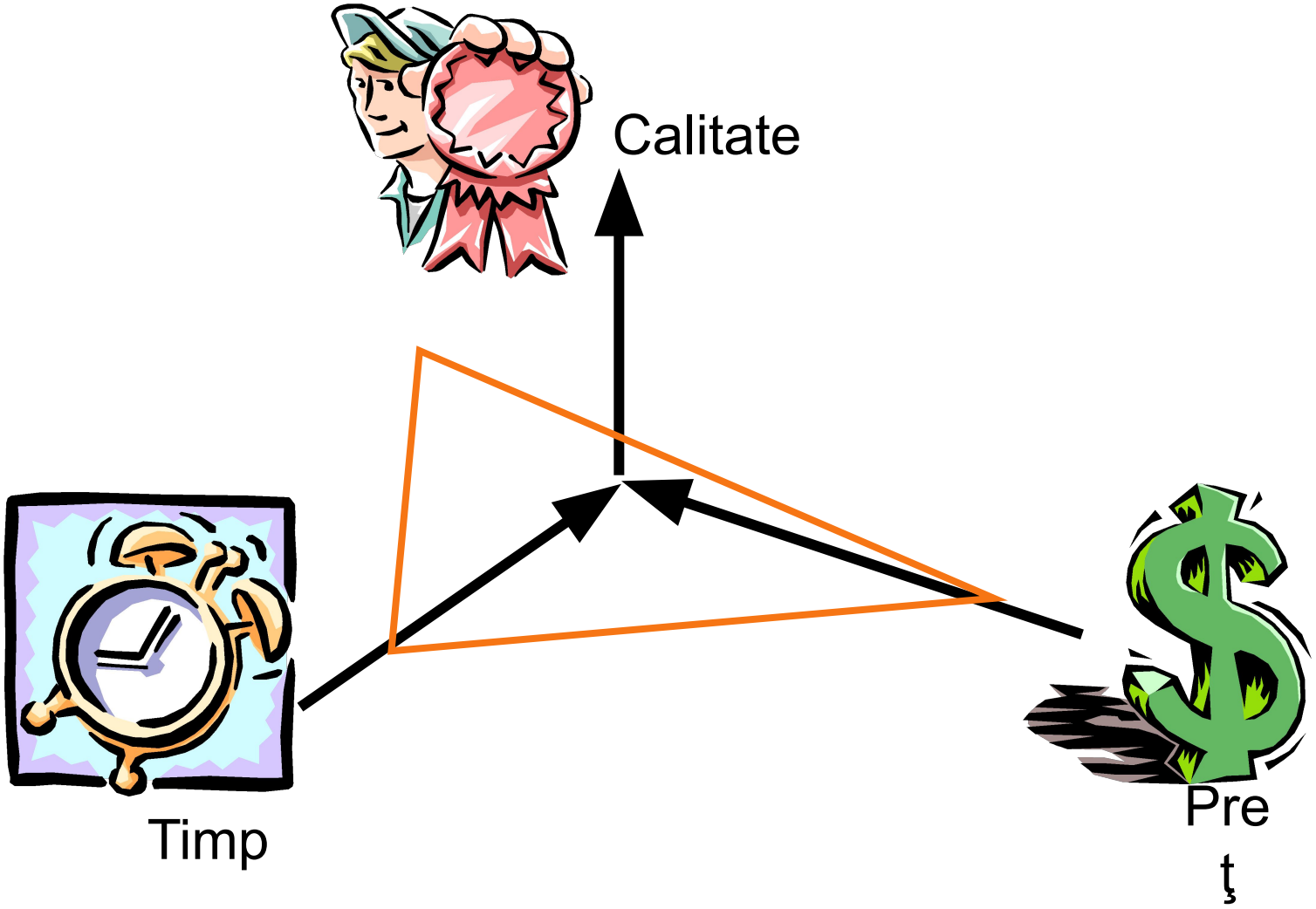
Introducere în Testare

Dilema Calităţii Software



Testare Software - Definiţie

"The process of exercising or evaluating

a system by manual or automated

means to verify that it satisfies specified

<u>requirements</u> 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 construcţiei produsului software
- Există documente de testare asociate la fiecare fază a dezvoltării

Care sunt Scopurile Testării?

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

Toate acestea se execută folosind *Metodologile de Implementare*

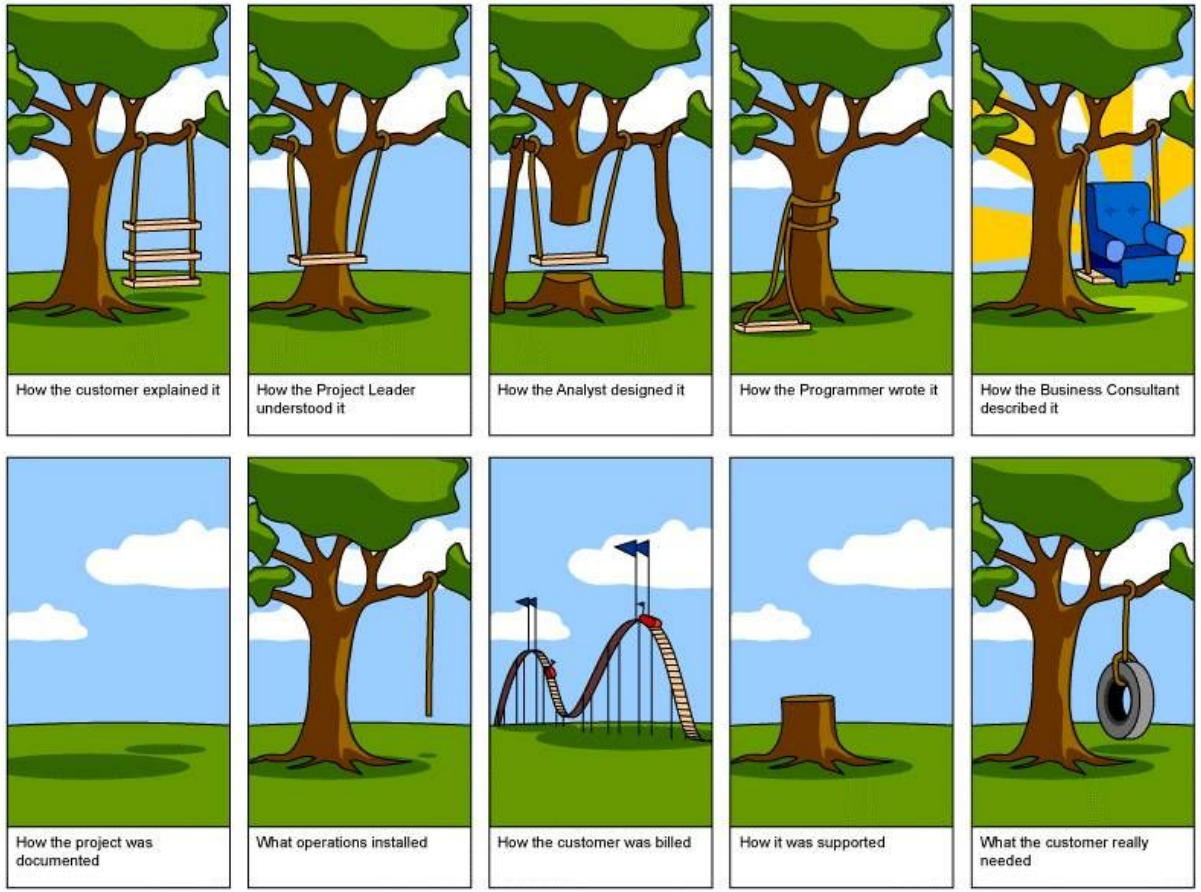
De ce avem “bugs” în Software?

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

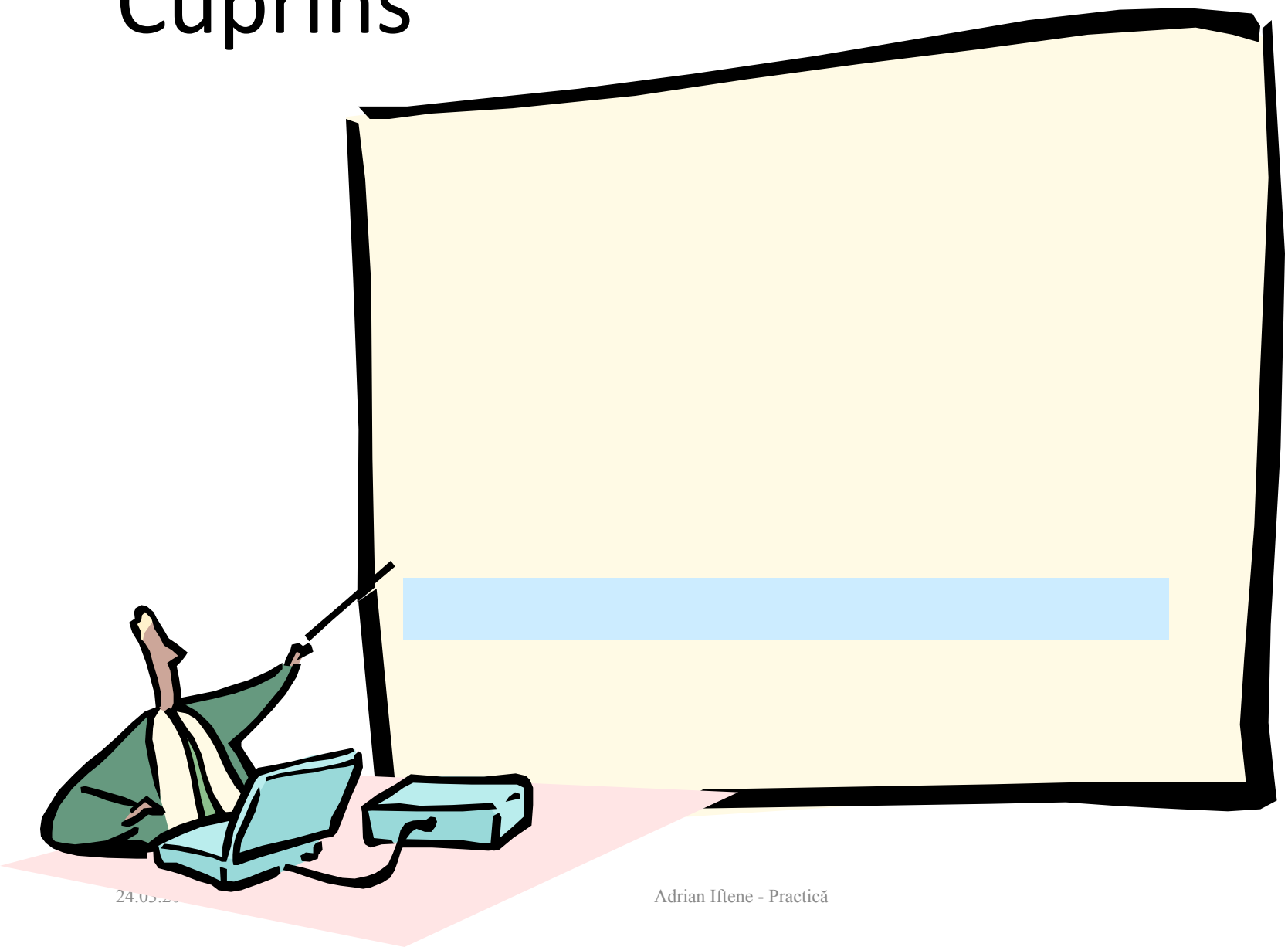
Comunicare Deficitară

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

Comunicare deficitară – În tratarea cerinţelor



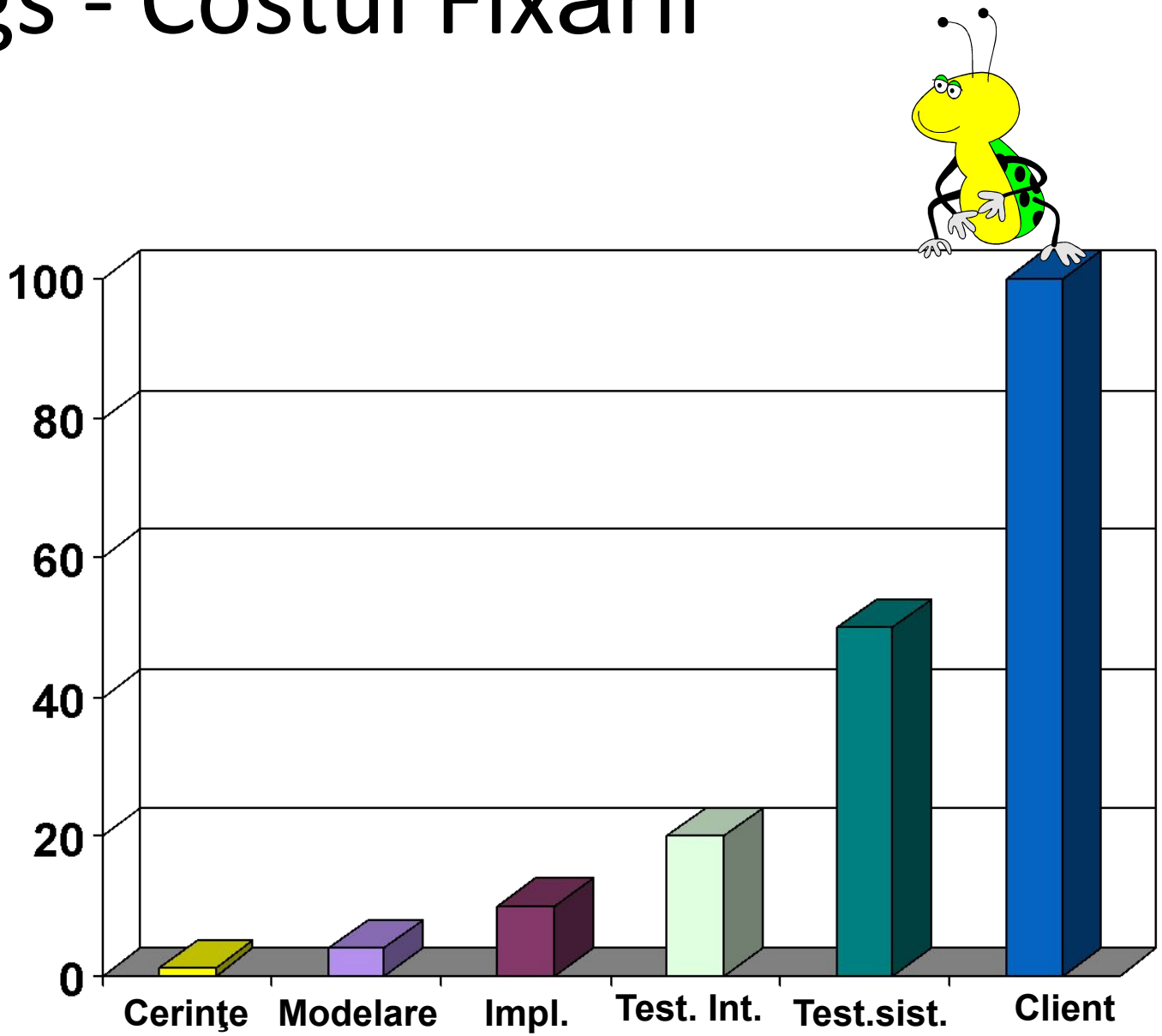
Cuprins



De unde vin Problemele Software?

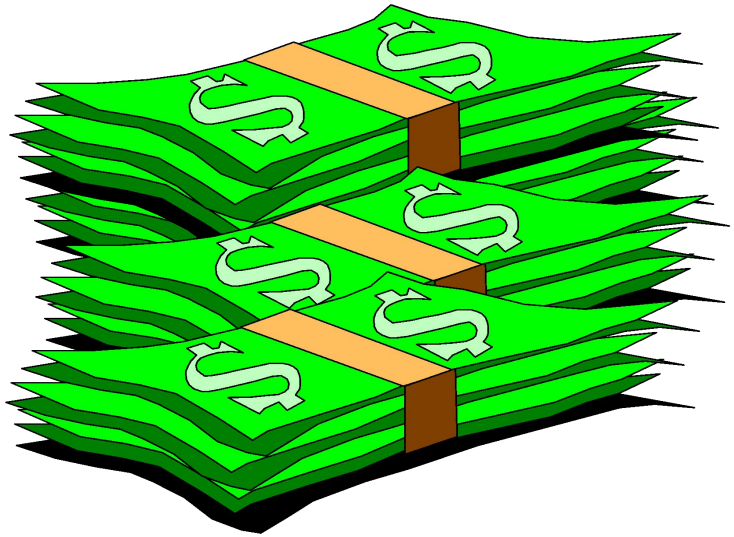
- Cerinţe definite incomplet 50%
- Modelare ambiguă sau insuficientă 30%
- Erori de programare 20%

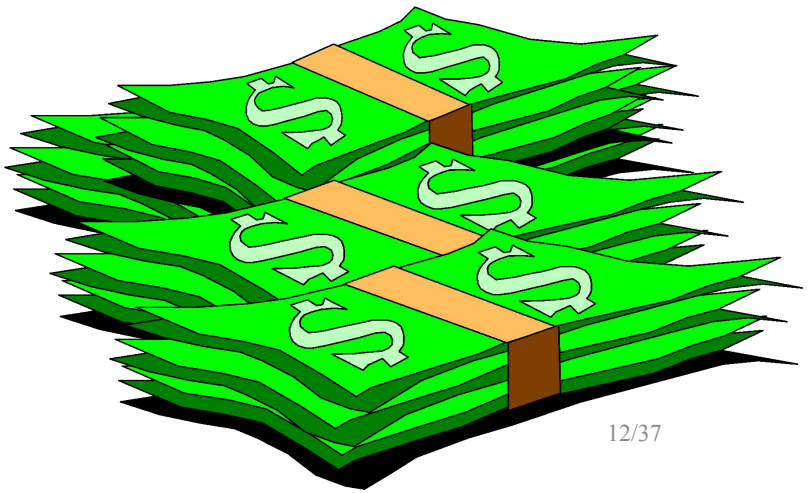
Bugs - Costul Fixării



Atenţie

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





Erori? Trebuie fixate cât mai Devreme Posibil

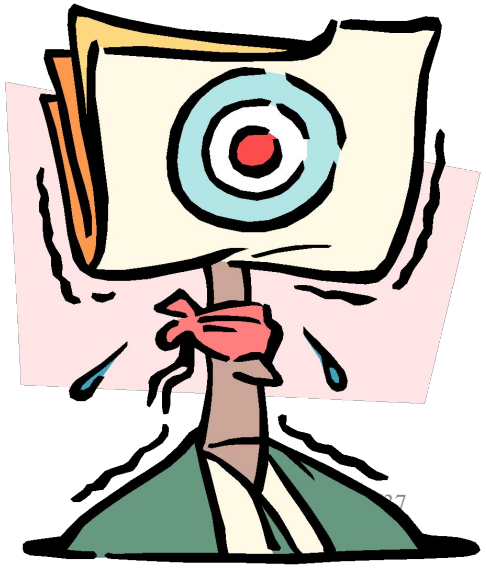
CERINŢE MODELARE IMPLEM. TESTARE CLIENT

Testare Profesională

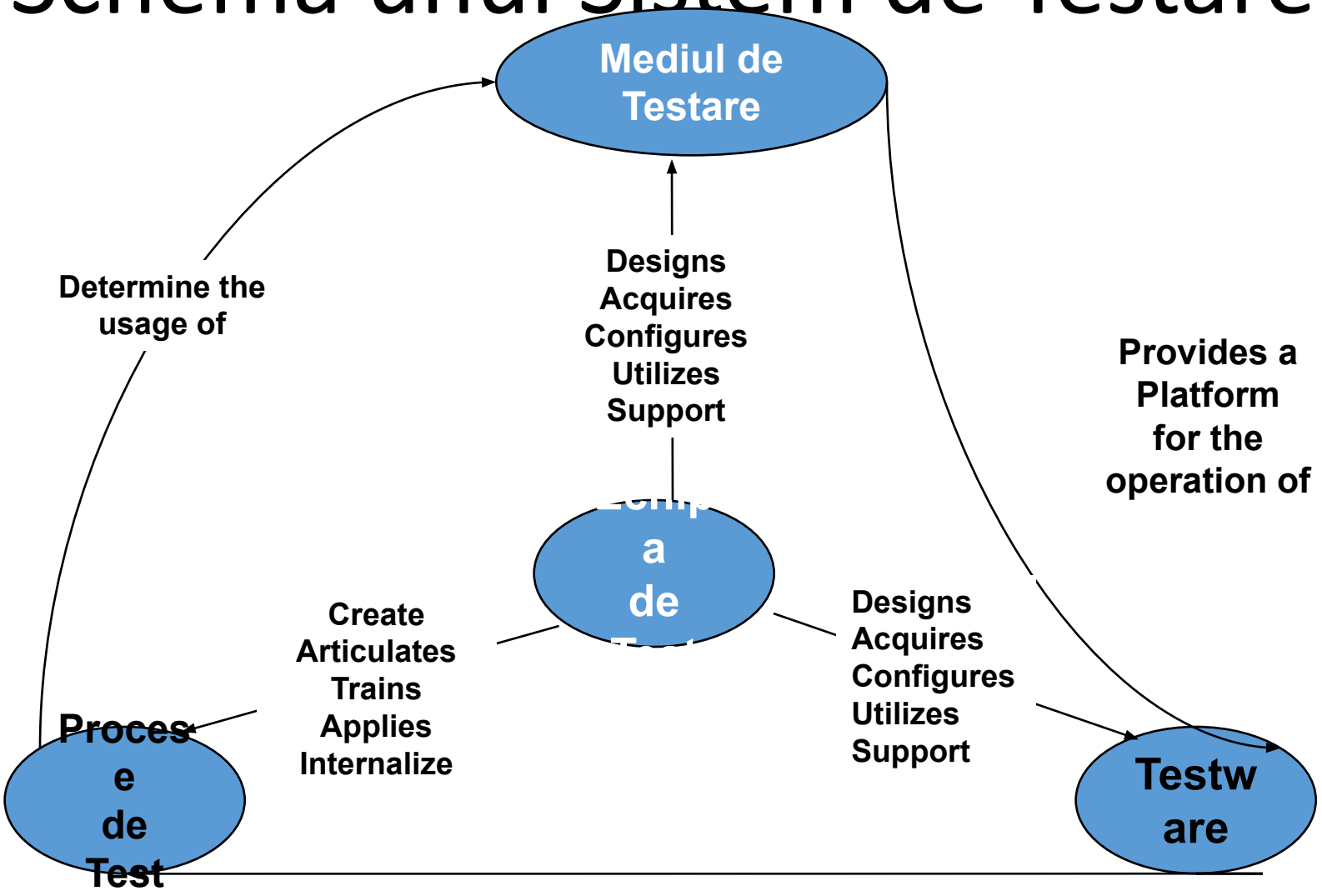
Profesionalismul în testare constă în abilitatea de a selecta numărul minim de cazuri de testare eficientă ce va fi capabil să verifice numărul maxim de funcţii ale sistemului.

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 şi majore
- Când timpul a expirat



Schema unui Sistem de Testare



Metodologii de Testare

Conţinut

- Diferenţa dintre testare SW şi debug SW
- Nivele de Test
- Clase de Test
- Conţinutul Testării
- Testare şi Dezvoltare SW

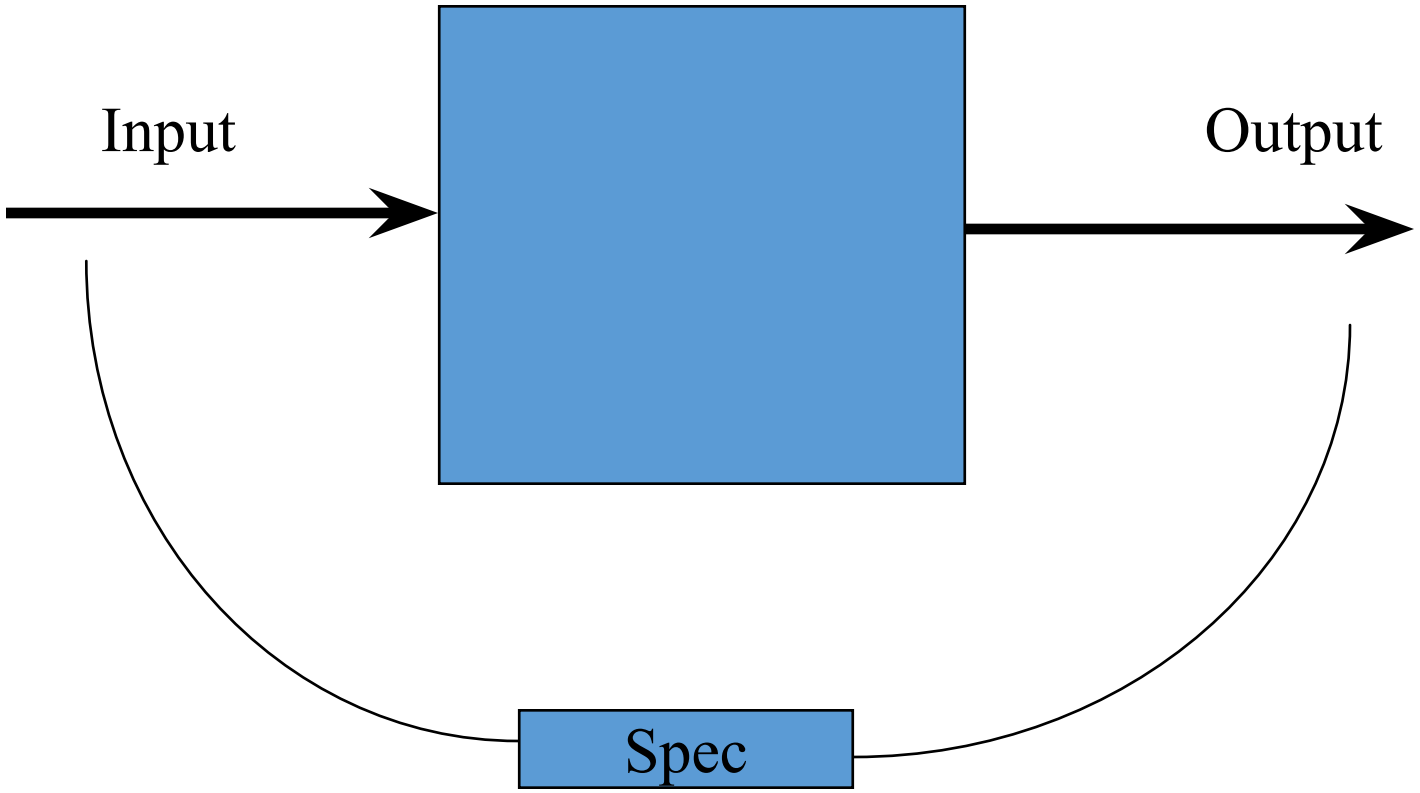
Diferenţa dintre testare software & debug

Testare	Debug
• Verificarea respectării cerințelor	• Verificarea validității secțiunilor
• De regulă e făcută de o entitate externă şi neutră	• E făcută de programator
• Este un proces planificat şi controlat	• E un proces aleator

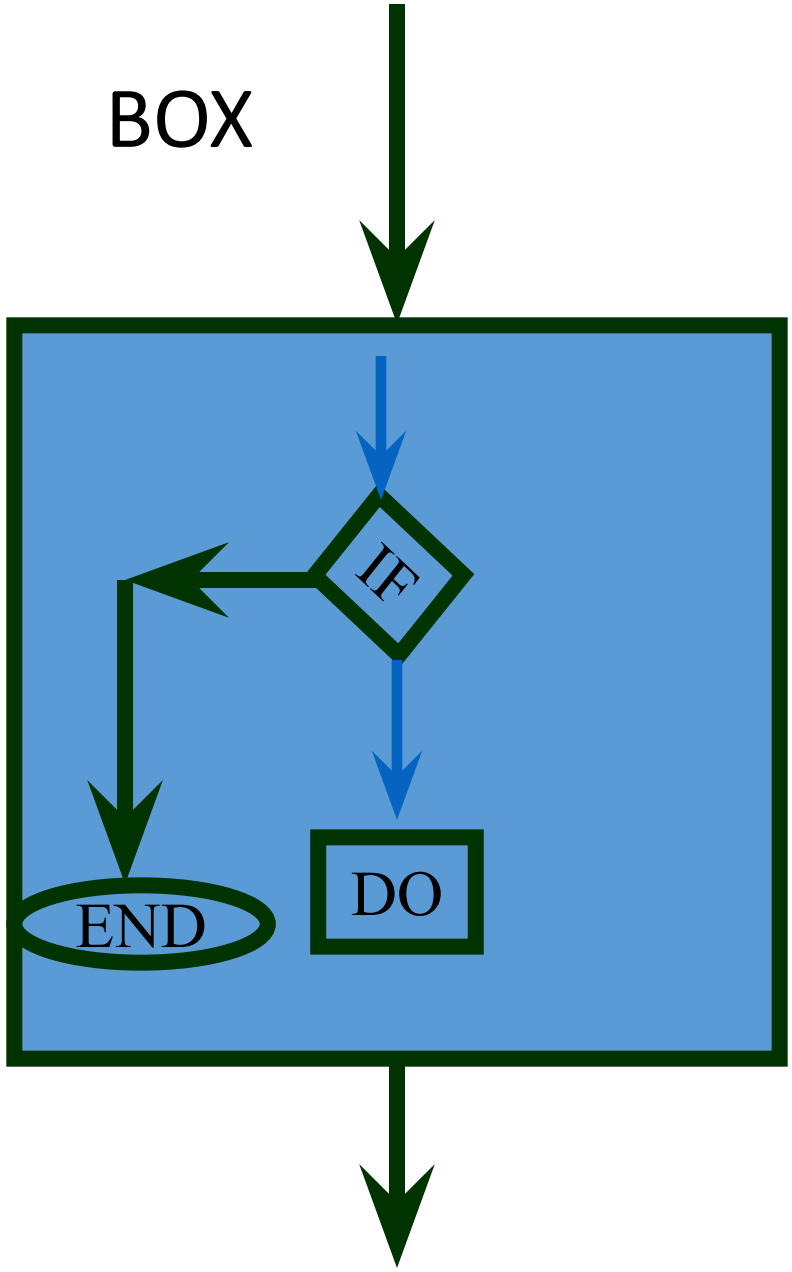
Nivele de Test

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

BLACK BOX



WHITE BOX



Unit Testing

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

Testare la Integrare

- **Testarea funcţionării unor module în acelaşi timp**
- **Testarea coexistenţei**
- **Se execută de către programatori sau de către testări analişti**
- **Testare pre-planificată**
- **Rezultatele se documentează**

Testare Automată vs Testare Manuală

- *Se găsesc rapid problemele*
- *Se câştigă 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 şi repetabilă*

- *Rezolvă problemele de interfaţă: 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 şi anevoioasă şi implică o cunoaştere 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=0411031

Coding Style – Motivaţie

- **Convenţiile de programare sunt importante deoarece:**
- **80% din timpul alocat unei componente software este întreţinere**
- **Foarte rar un produs software este întreţinut pe toată durata folosirii lui de către aceeaşi persoană**
- **Convenţiile de cod îmbunătăţesc lizibilitatea produsului, şi permite inginerilor software să înţeleagă rapid un program nou**

Coding Style - Cerinţe

- **Folosirea fără rezerve a Comentariilor: ce fac procedurile, ce reprezintă variabilele, explicarea paşilor 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 in calcul şi originalitatea ideii precum şi alegerea corespunzătoare a tehnologiei de implementare. Acolo unde este cazul se ia in considerare munca necesara implementării interfeţei si calitatea rezultatului.**
Fişa cerinţelor, raportul final, activitate	**A: 2p B: 1p C: 0p**	**Se punctează modul în care a fost realizată fişa cerinţelor şi felul în care studentul a interacţionat cu conducătorul de laborator ("clientul"). Un punctaj bun se acordă dacă clientului îi este clar cu ce se ocupă proiectul şi i-a fost prezentată evoluţia proiectului.**
Stil de programare	**A: 1p B: 0.5p C: 0p**	**Se punctează aderarea la stilul de programare menţionat în fişierul raport.html, denumirea auto-descriptivă a variabilelor etc. In aceasta categorie intra si folosirea adecvata a unui sistem de generare automata a documentaţiei (javadoc, doxygen etc.)**
Proiectare şi 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 unităţilor de testare automată în cadrul proiectului (JUnit, cppunit, NUnit, phpunit ...)**