Ingineria Programării

Cursul 12 – 23 mai

Cuprins

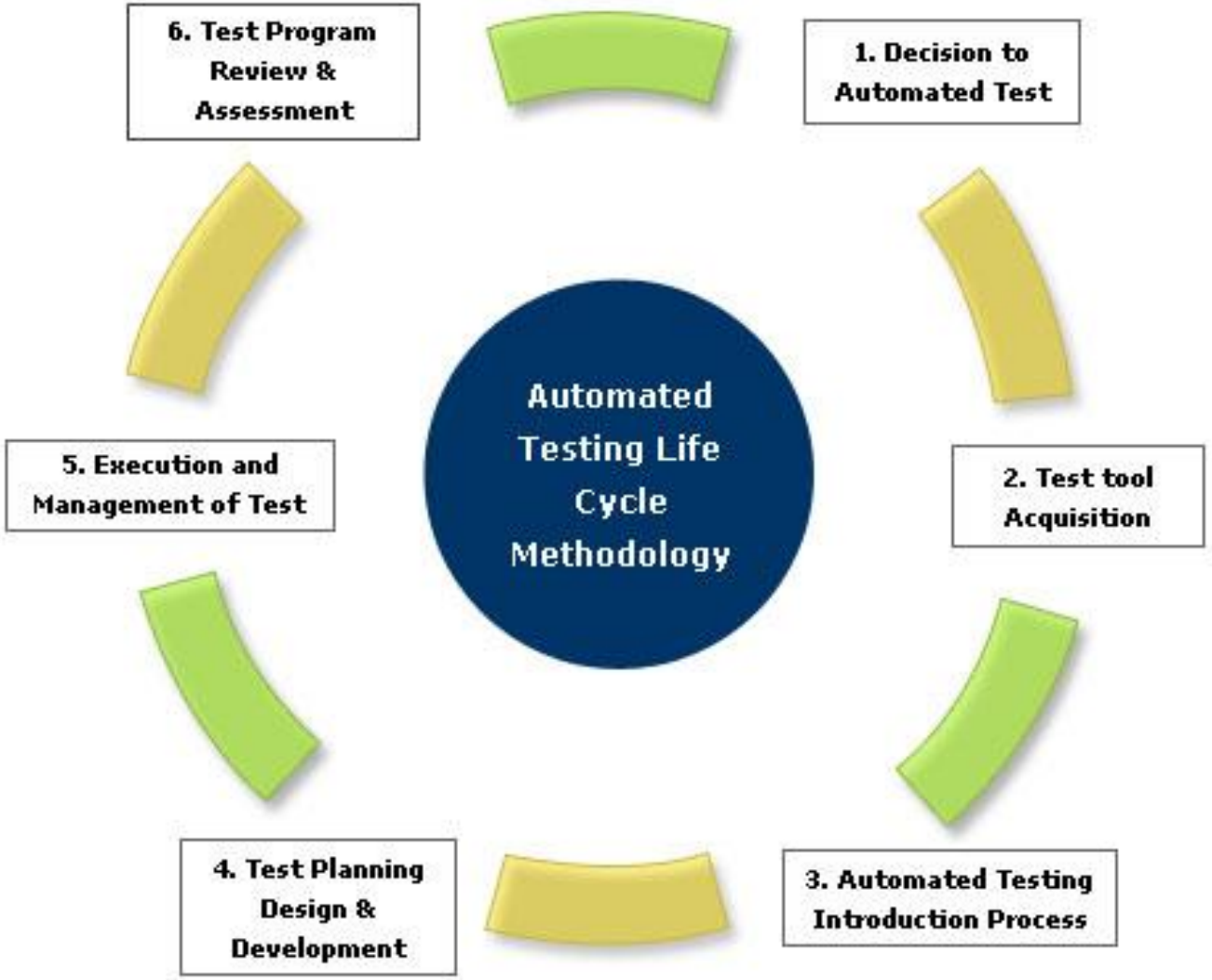
- Previous courses…
  - Testing
  - Test Automation
  - Software Bug
  - Testing cycle
- Program Quality
- Metrics
- Copyright

R: Manual Testing

- *How, Who, When, Where, Results*

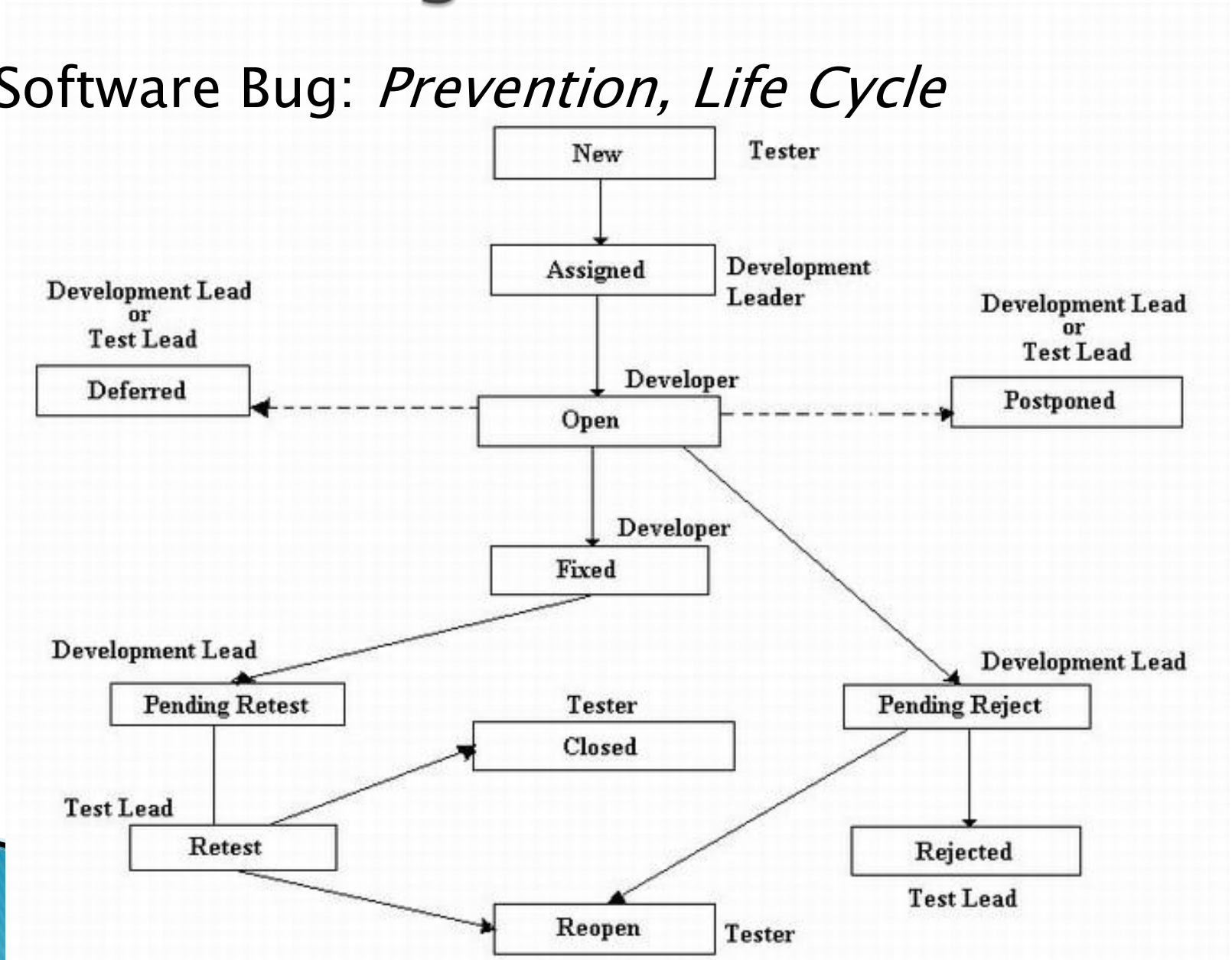
Test Automation

- Test Automation: *How, Who, When, Results*

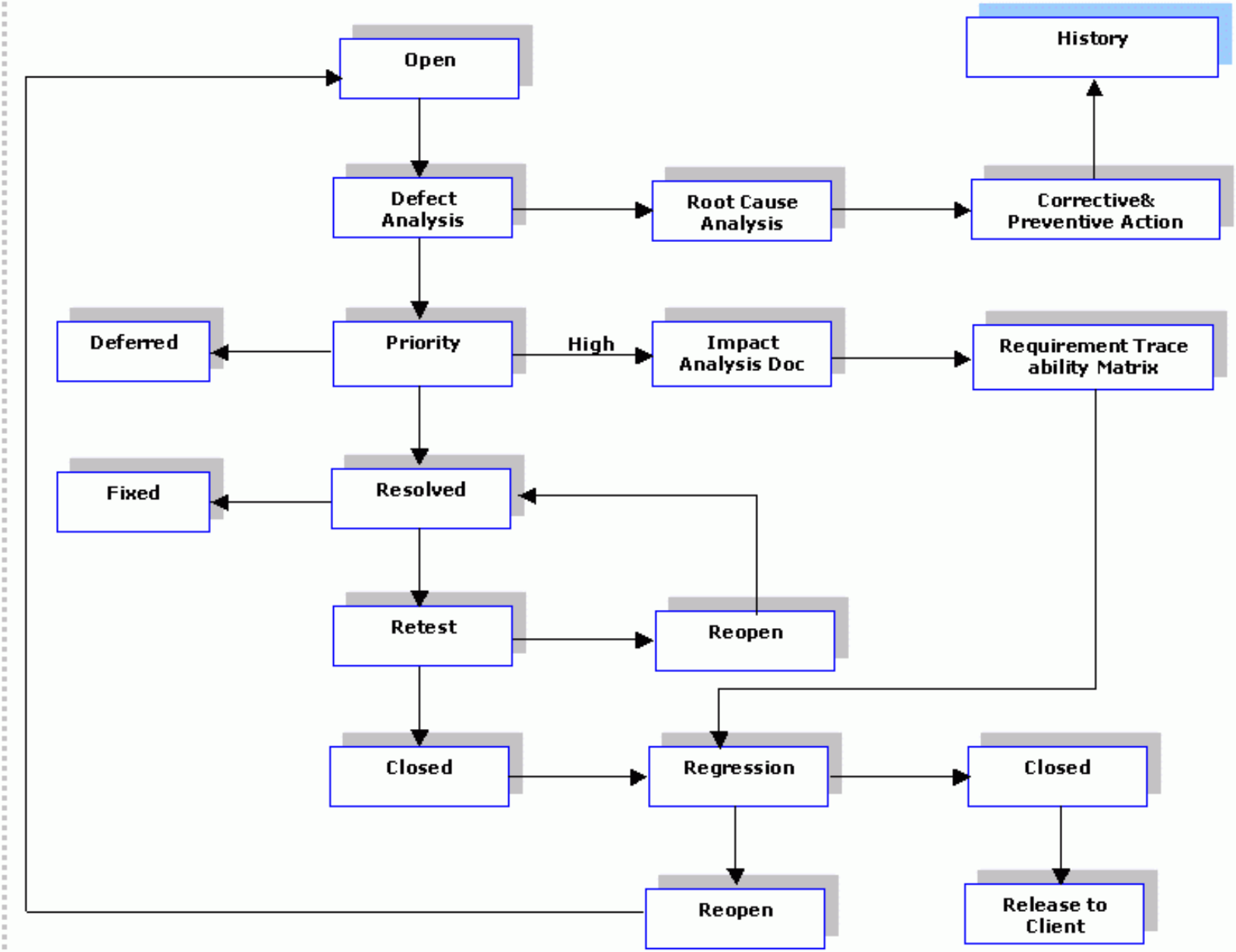


Software Bug

- Software Bug: *Prevention, Life Cycle*



Bug Life Cycle



R: Testing Cycle

- Testing Cycle: *Persons involved, Tasks*

Assessing quality

- How do we measure the quality of an item?
  - **Construction quality** (how well it is built, whether the raw material has flaws, etc ...)
  - **Design quality** (comfort, elegance...)
  - A combination between quality of desogn and construction (sturdiness...)

- In general, we can say that chair A is better than chair B regarding some particular aspect, but it is usually difficult to say by how much.

Calitatea unui program (1)

- Nu se examinează calitatea construcției (=> unicitate între toate disciplinele inginerești)
- Toate atributele calității se referă la design.
- Dacă ne referim la valori estetice:
  - Programele sunt în mare parte invizibile, iar valorile estetice contează doar pentru părțile vizibile
  - În afară de interfață, singurele aspecte observabile la un program sunt:
    - Notațiile folosite pentru design și scrierea codului
    - Comportamentul programului atunci când interacționează cu alte entități.

Calitatea unui program (2)

- Atunci când vorbim despre calitatea unui program trebuie:
  - **Să definim atribute ale calității** care ne interesează;
  - Să căutăm **modalități de măsurare** a lor;
  - Să putem da o **reprezentare a designului**;
  - **Să scriem specificații** care să ghideze programatorii (calitățile designului să fie respectate și de implementare).

Codul sursă

- Codul care implementează un design este o reprezentare a acelui design.
- Asigurarea calității făcută după scrierea codului este un proces costisitor și e posibil să fie inutil.
- De regulă doar se ține cont de modul în care e scris codul (coding style, design patterns, adaptability, maintenance, reuse (cuplaj, coeziune), security)

Calitatea unui program (3)

- Măsoară cât de bine se potriveste un program mediului său.
- Ia în calcul aspecte de tipul:
  - Programul funcționează;
  - Programul face ceea ce trebuie să facă;
  - Programul este sigur;
  - Programul poate fi adaptat pe măsură ce nevoile se schimbă.
- Toate măsurile referitoare la calitate sunt relative!!!

Concepte ale calității programelor

- Siguranță (safety)

- Eficiență (efficiency)

- Întreținere (maintenance)

- Uzabilitate (usability)

Siguranța

- Este programul **complet**, **consistent** și **robust**?
  - **completitudine** – tratează toate intrările posibile;
  - **consistență** – se comportă întotdeauna așa cum este așteptat;
  - **robustețe** – se comportă bine în situații anormale (ex. lipsa resurselor, lipsa conexiunii, etc.)

Eficiența

- Programul utilizează într-un mod eficient resursele? (procesor, memorie, rețea...)
- Eficiența este întotdeauna mai puțin importantă decât siguranța: *Este mai ușor să facem un program sigur să fie eficient decât un program eficient să fie sigur*

Întreținere

- Cât de ușor poate fi modificat design-ul ulterior?

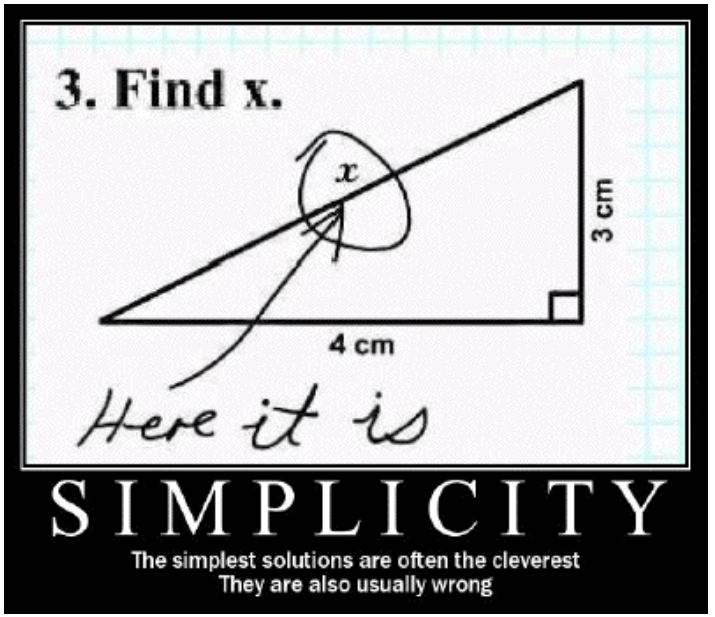
- Tipuri de întreținere:
  - **Corectivă**: eliminarea erorilor;
  - **Perfectivă**: adăugarea de funcționalități care ar fi trebuit să fie oferite;
  - **Adaptivă**: actualizarea programului când se schimbă cerințele.

Uzabilitate

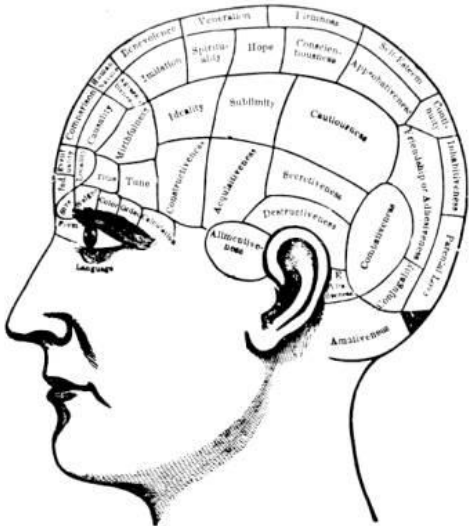
- Cât de usor este programul de învățat și de folosit?

Atribute măsurabile

- Simplitate



- Modularitate



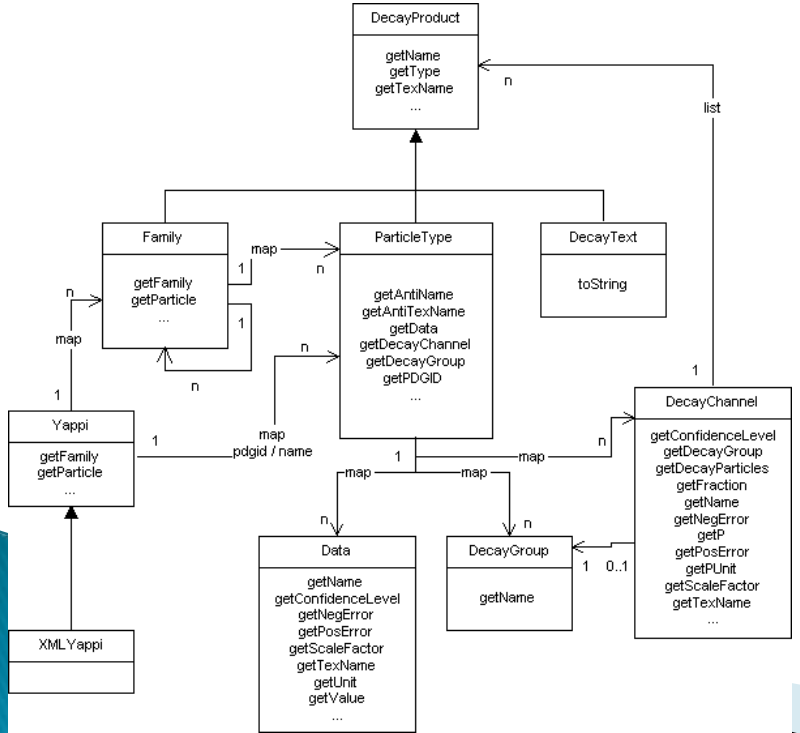
Simplitate

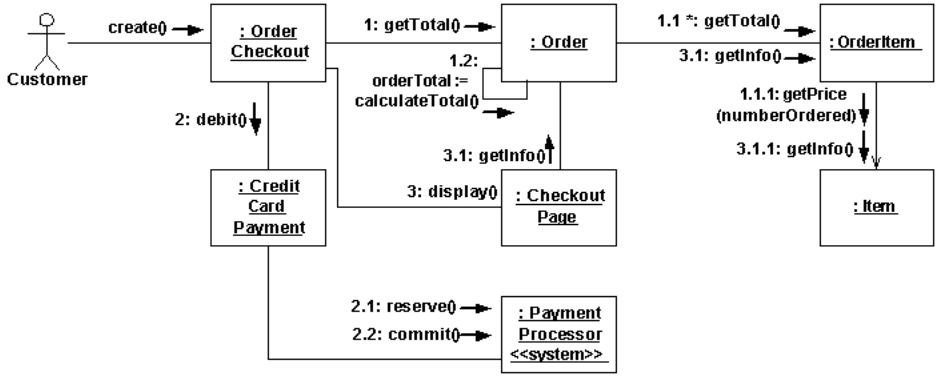
- Este măsura inversă a complexității.

- Aspecte ale complexității:
  - Complexitatea **fluxului de control**: numără căile posibile de execuție ale unui program
  - Complexitatea **fluxului de informații**: numără datele care sunt transmise în program
  - Complexitatea **înțelegerii**: numără identificatorii și operatorii folosiți

Modularitate

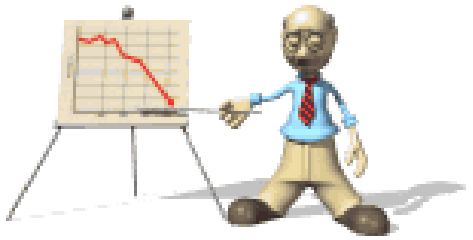
- Poate fi măsurată uitându-ne la:
  - **Coeziune**: cât de bine lucrează împreună componentele unui modul.
  - **Cuplaj**: gradul de interacțiune între module.





Motivații pentru metrici

- Efectuăm măsurători pentru
  - a înțelege
  - a controla
  - a prevedea



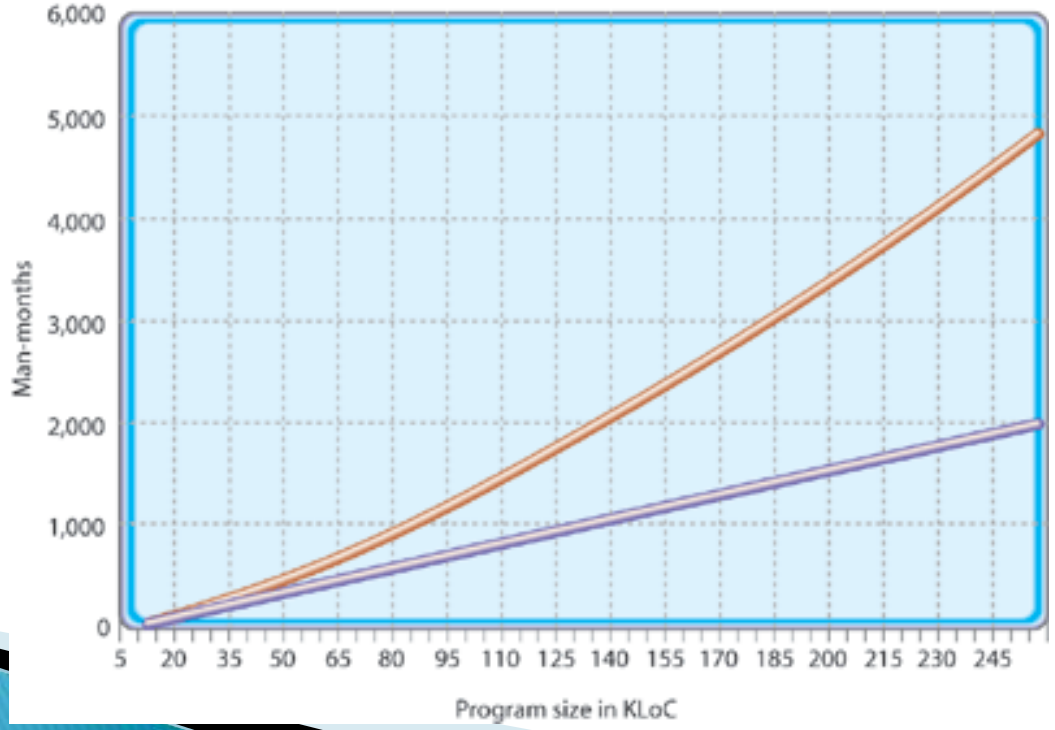
- *Când poți să măsori lucrul despre care vorbesti și să exprimi aceasta în numere, atunci știi ceva despre acel lucru. Dar când nu poți să îl măsori, când nu poți să îl exprimi în numere, cunostințele tale sunt slabe și nesatisfăcătoare; ele pot fi începutul cunoasterii, dar mai nimic nu este încă făcut pentru a ajunge la stadiul de știință. (Lord Kelvin)*

Ce se dorește a fi măsurat

- Dimensiunea unui program
- Complexitatea unui program
- Fiabilitatea unui program
- Timpul necesar dezvoltării unui program
- Alocarea resurselor necesare dezvoltării unui program
- Productivitatea muncii
- Costurile de dezvoltare

Metrici de bază

- KLOC: Kilo Lines Of Code (mii linii de cod)
- Effort, PM: Person – Month (Om – lună).



COCOMO

- COnstructive COst MOdel (Boehm 1981)
- Folosit pentru evaluarea costurilor
- Trei nivele de rafinare a predicțiilor:
  - COCOMO de bază
  - COCOMO intermediar
  - COCOMO detaliat

COCOMO de bază

- Formula pentru efortul necesar dezvoltării în funcție de numărul de linii de cod

- **Effort Applied** = a_b (KLOC)^{bb} [man-months]
- **Development Time** = c_b (Effort Applied)^{db} [months]
- **People required** = Effort Applied / Development Time [count]

- a_b , b_b, c_b, d_b – constante ce depind de tipul proiectului

COCOMO 2

- Propus de Boehm în 1995
- Ia în calcul tehnicile moderne de dezvoltare apărute
  - Prototipizare
  - Dezvoltarea pe componente
  - 4GL (fourth generation language)
- Oferă posibilitatea de a face estimări încă din primele faze ale dezvoltării

COCOMO 2: Prototipizarea inițială

- Surprinde efortul necesar realizării unui prototip al aplicației
- Se bazează pe numărul de puncte obiect (NOP)
- Formula de calcul a efortului:

$$PM = (1 - P_{reuse}) \frac{NOP}{PROD}$$

NOP

- Se inventariază ecranele ce trebuie afișate
  - Simple: 1
  - Complexe: 2
  - Foarte complexe: 3
- Se inventariază rapoartele ce trebuie generate
  - Simple: 2
  - Complexe: 5
  - Foarte complexe: 8
- Fiecare modul în limbaj de nivel mai jos (ex. 3GL): 10
- Suma punctelor obținute reprezintă numărul total de puncte obiect ale programului.

COCOMO 2: după proiectare

- Se estimează numărul total de linii de cod (ESLOC)
- Factori luați în calcul
  - Volatilitatea cerințelor
  - Gradul posibilității de reutilizare a codului

COCOMO 2: influența asupra costurilor

- Atributele produsului
  - Siguranța produsului; Complexitatea modulelor sistemului; Dimensiunea documentației cerute; Dimensiunile bazei de date utilizate; Procentajul cerut de componente reutilizabile

- Atributele sistemului de calcul
  - Constrângeri privind timpul de execuție; Volatilitatea platformei pe care se face dezvoltarea; Constrângeri de memorie

COCOMO 2: influența asupra costurilor (cont.)

- Atributele personalului
  - Capacitatea analiștilor; Capacitatea programatorilor; Continuitatea personalului; Experiența analistului în domeniul problemei; Experiența programatorilor în domeniul problemei; Experiența în limbajele și instrumentele folosite

- Atributele proiectului
  - Utilizarea intrumentelor; Gradul de lucru în echipe aflate la distanță și calitatea comunicării între echipe; Comprimarea planului de dezvoltare

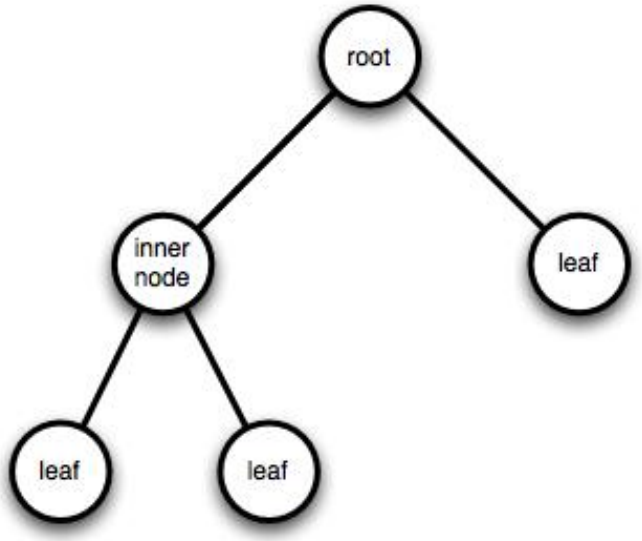
Distribuția forței de muncă în timp

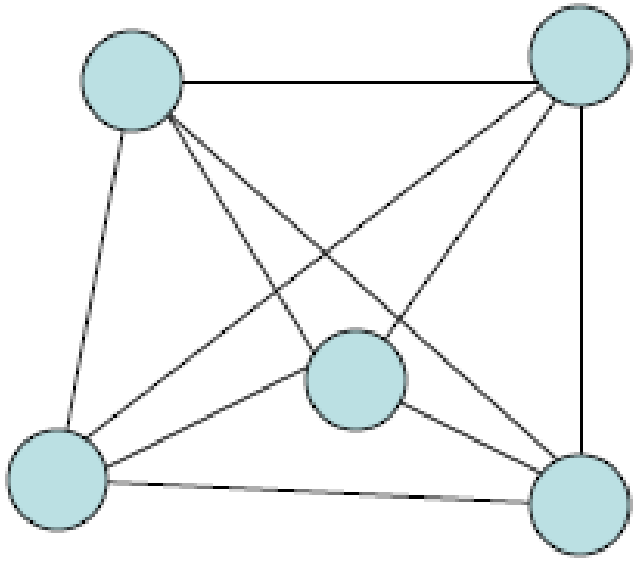
- 20 om-lună. E corect calculul de mai jos???
  - 20 oameni muncesc 1 luna
  - 4 oameni muncesc 5 luni
  - 1 om muncește 20 luni
- Productivitatea individuală scade atunci când echipa de dezvoltare crește
  - Comunicare suplimentară
  - La adăugarea de noi membri, productivitatea inițială scade
- *Creșterea numerică a forței de muncă la un proiect rămas în urmă are ca efect rămânerea și mai în urmă a proiectului.* (Legea lui Brooks)

Distribuția forței de muncă în timp (2)

- Pentru o echipă cu P persoane putem avea între P-1 și P(P-1)/2 canale de comunicație

- Fiecare canal induce o pierdere de eficiență





Cum *NU se calculează costurile* și *NU se fac planificări*

- Avem 12 luni să terminăm treaba, deci treaba va dura 12 luni. Legea lui Parkinson: *munca se întinde pe timpul avut la dispoziție*.
- Știm că competitorul a cerut $1.000.000. Noi cerem $900.000.
- Știm că bugetul clientului pentru produs este de $500.000. Atât ne costă și pe noi dezvoltarea.
- Programul va dura 1 an, dar voi spune că va dura 10 luni. Ce-o să mai conteze 2 luni peste planificare...

Probleme la folosirea metricilor

- Lipsa de acuratețe
- Rezistența din partea angajaților
- Folosirea lor în alte scopuri decât au fost create
- Disensiuni în cadrul echipei de dezvoltare

Copyright

- Drepturile de autor reprezintă ansamblul prerogativelor de care se bucură autorii cu referire la operele create; instituția dreptului de autor este instrumentul de protecție a creatorilor și operelor lor
- *Copyright gives the creator of an original work exclusive right for a certain time period in relation to that work, including its* ***publication, distribution and adaptation****; after which time the work is said to enter the public domain.*

Exclusive rights

- Several exclusive rights typically attach to the holder of a copyright:
  - to produce copies or reproductions of the work and to sell those copies (mechanical rights; including, sometimes, electronic copies: distribution rights)
  - to create derivative works (works that adapt the original work)
  - to perform or display the work publicly (performance rights)
  - to sell or assign these rights to others
  - to transmit or display by radio or video (broadcasting rights)

Evaluare proiecte

Șeful de grupa prezintă ce a lucrat fiecare membru al grupei (clase și module implementate, resurse, documentație, etc.). Evaluare proiect integrat:

- L9 + L10 – 40 puncte – implementare
- L11 – integrare
  - 10 puncte integrare cu modulul anterior si daca functioneaza
  - 20 de puncte daca functioneaza proiectul la nivel de grupa
- L12 – testare (10 puncte de metoda de test de om)
- L13 – evaluare
- **0 puncte fără integrare**

Cei care nu implementează nimic sau încurcă celelalte subgrupe în realizarea proiectului, primesc punctaj negativ…

Examen scris

- Tematici: IP, logică, cunoștințe generale, etc.
- ~15 subiecte, ~30 de minute
- Răspunsurile scurte la obiect, cuvinte cheie

- Atenție! Vor fi subiecte a căror nerezolvare va duce la acumularea unui punctaj negativ! Aceste subiecte vor fi marcate pe teză cu punctajul sub forma [-10, 10] (lucrurile de bază…)
- Data: soon™

Examen scris

Links

- **Bug Life Cycle:** http://www.buzzle.com/editorials/4-6-2005-68177.asp, http://qastation.wordpress.com/2008/06/13/process-for-bug-life-cycle/
- **COCOMO:** http://en.wikipedia.org/wiki/COCOMO
- Curs 12, Ovidiu si Adriana Gheorghies: http://www.info.uaic.ro/~ogh/files/ip/curs-12.pdf