

Ingîneria programelor (Software engineering)

INTRODUCERE

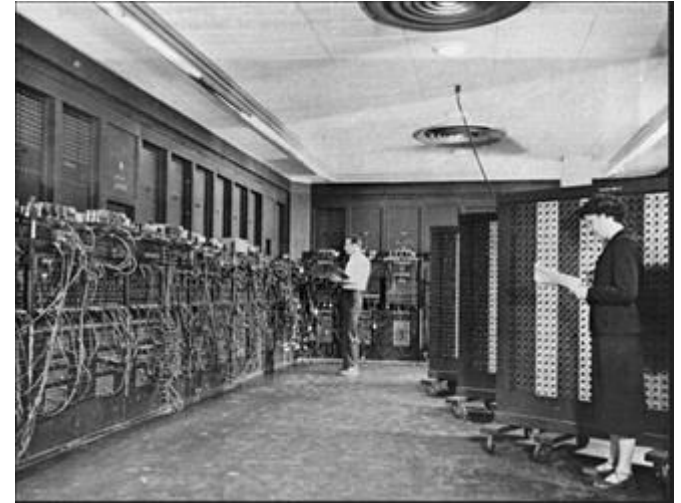
Prof. univ. dr. ing. Florica Moldoveanu

UPB, Automatică și Calculatoare
2020-2021

Inceputurile

❖ **Primul calculator electronic digital programabil:** ENIAC (Electronic Numerical Integrator And Computer) -1946

- construit la universitatea Pensilvania
- a functionat neintrerupt in perioada 1946-1955
- cântărea 27 t, avea aprox. 2,6 m x 0,9 m x 26 m, ocupa 63 m², și consuma 150 kW
- programarea + introducerea datelor:
prin switch-uri de pe panourile functionale



<http://en.wikipedia.org/wiki/ENIAC>

❖ **Primul calculator electronic digital creat in Romania:** 1957, la IFA (Institutul pentru Fizica Atomica)

❖ **Primul dispozitiv de intrare:** cititor de cartele



❖ **Primul limbaj de programare de nivel inalt:** FORTRAN

- primul compiler de Fortran: 1954-1957, pentru IBM 704
- ultimele versiuni standard ale limbajului FORTRAN: 2003 (cu suport pentru POO), 2008, 2018.

Criza software(1)

Anii 1970 – “Criza software”

Cauzele:

- Evolutia rapida a tehnologiilor hardware si ieftinirea echipamentelor de calcul.
 - Intelegerea beneficiilor folosirii calculatoarelor: rapiditatea obtinerii rezultatelor, acuratetea rezultatelor, s.a.
- 
- Cresterea cerințelor de dezvoltare de programe, de dimensiune și complexitate din ce in ce mai mari.
- 
- Necesitatea cresterii productivitatii activitatilor de dezvoltare software (timp prea mare de dezvoltare) și a imbunatatirii calitatii produselor software (erori in functionare)
 - Insuficient personal calificat pentru satisfacerea acestor cerinte.

Criza software(2)

Concluzii:

Metodele de dezvoltare software existente la acea vreme – inadecvate cerintelor

- Efortul de dezvoltare creste mai mult decat liniar raportat la dimensiunea programului.
- Programele trebuie sa poata fi usor de inteles si de adaptat de persoane diferite de cele care le-au dezvoltat.
- Programele nu sunt entitati statice, ele evolueaza in timp datorita schimbarii cerintelor si a mediului de utilizare.

Dezvoltarea de software trebuia sa devina o industrie de sine statatoare.

Grady Booch: The net result is that building and maintaining software is hard and getting harder; building **quality software** in a **repeatable** and **predictable** way is harder still.

Software Engineering(1)

➤ A devenit necesara o **disciplina** care sa furnizeze cadrul pentru **construirea de software de calitate în mod repetabil și predictibil.**

➤ **Software Engineering (Ingineria programelor)**

Termenul: dezvoltarea programelor trebuie să fie fundamentată teoretic și ghidata de metode validate în practică, la fel ca alte domenii ingineresti.

➤ **Scopul disciplinei:** definirea de tehnici de “fabricatie” justificate de teorie sau de practica →
poate fi software-ul tratat ca alte produse ingineresti?

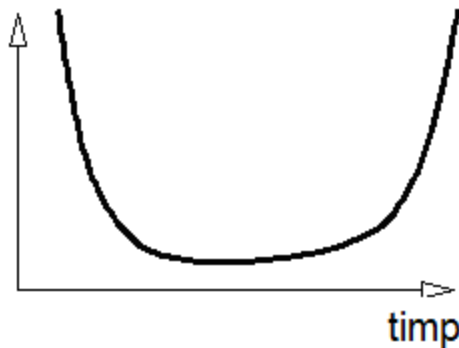
Software-ul se deosebeste de alte produse, care sunt fabricate:

- nu este un produs fizic
- un produs hardware este dezvoltat (conceptie, proiectare, constructie prototip, testare prototip), apoi fabricat in serie
- un produs software este doar dezvoltat nu si fabricat; nu exista un proces de fabricatie software
- programele nu pot fi complet “asamblate” din componente – orice program nou trebuie sa raspunda la cerinte specifice

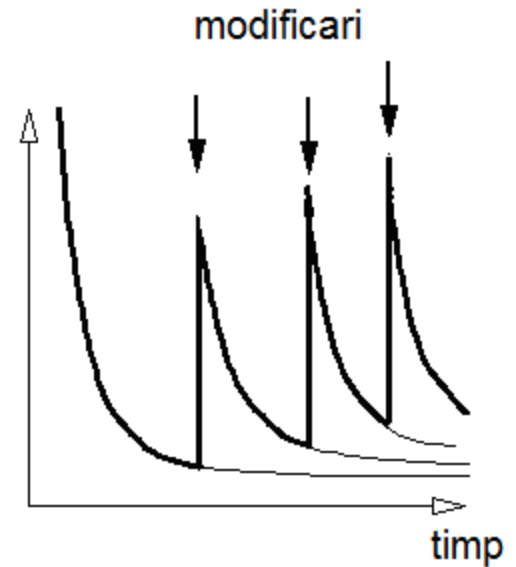
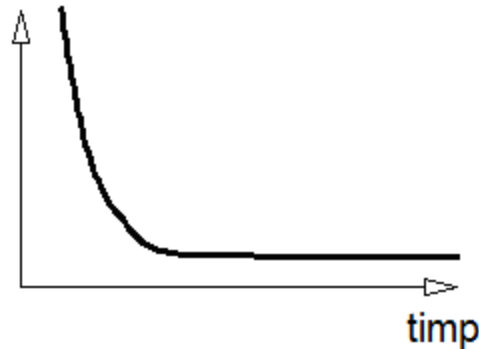
Software Engineering(2)

- un produs hardware nu poate fi modificat pe parcursul utilizarii sale pentru a fi adaptat la cerinte noi
- software-ul nu “îmbătrânește”

Rata caderilor unui echipament (hardware)



Rata caderilor unui program



Software Engineering si crizele software (1)

- Prima criza software: perioada '60 – '70
 - Cauza: Necesitatea de a dezvolta programe de dimensiuni mari, intr-un timp mai scurt, mai fiabile
 - Rezolvare: Trecerea de la limbaje de asamblare la limbaje de programare de nivel inalt (Fortran, C – începutul anilor '70)
- A 2-a criza software: perioada '80 – '90
 - Cauza: Incapacitatea de a dezvolta programe de dimensiuni foarte mari, complexe, cu participarea unui numar mare de programatori
 - Rezolvare:
 - Trecerea la limbaje de programare orientate obiect (C++, Java, C#)
 - Instrumente de dezvoltare mai puternice
 - Noi metodologii de dezvoltare: șabloane de proiectare, sabloane arhitecturale, revizia specificatiilor, a testelor, a codului
- A 3-a criza software: dupa 2010
 - Cauze:
 - Performanta procesarii secventiale a devenit nesatisfacatoare
 - Necesitatea stocarii si procesarii de volume de date mari
 - Rezolvare: Noi modele de programare, paralelizare, noi limbaje de programare, standardizare, instrumente pentru paralelizare si testare, ș.a.

Software Engineering - definitie

- Una dintre definițiile termenului “Software Engineering” (IEEE Standard Glossary of Software Engineering Terminology,” IEEE std 610.12-1990):

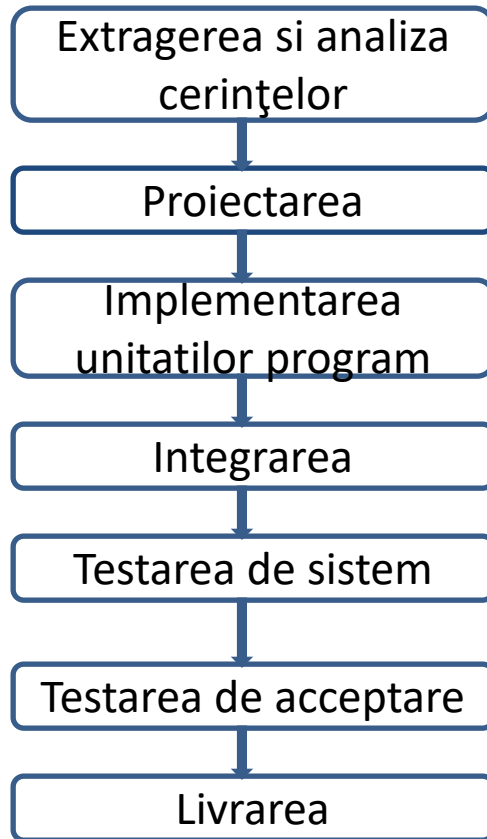
Software Engineering este:

“Aplicarea unei abordări sistematice, disciplinate și măsurabile în dezvoltarea, operarea și întreținerea software-ului, adică aplicarea ingineriei pentru software. De asemenea, studiul unor asemenea abordări.”

- **Produs software =**
cod sursa, cod executabil, biblioteci
+
documentații (de specificare a cerintelor, de proiectare, de instalare, de utilizare)

Activități în dezvoltarea produselor software

- Activități tehnice:



Specifici **CE** îți propui sa faci, pentru a satisface cerintele clientului/viitorilor utilizatori

Decizi **CUM** vei face

Creezi mai întâi părțile

Asamblezi părțile pentru a obtine produsul

Verifici că produsul rezultat functioneaza asa cum ti-ai propus (cum ai specificat)

Verifici împreună cu clientul și utilizatori finali ca produsul satisface asteptarile lor

Instalezi produsul în mediul său de operare

- Activități de asigurare a calitatii

- Activități de management al proiectului de dezvoltare software

Extragerea si analiza cerintelor (1)

- **Extragerea cerințelor:** identificarea si definirea cerintelor utilizator
- **Cerintele utilizator**
 - Descriu punctul de vedere al utilizatorilor: CE doresc viitorii utilizatori de la viitorul produs.
 - Sunt cerinte de: functionare, performanta, securitate, interfata utilizator, interfete de comunicatie, s.a.
 - Definite în ***Documentul cerintelor utilizator (User Requirements Document) – URD***
 - In URD cerintele sunt exprimate in limbaj natural si folosind notatii grafice usor de inteles de client si viitorii utilizatori.
 - URD devine **parte din contractul cu clientul**.
 - URD este documentul de referință pentru testarea de acceptare.

Extragerea si analiza cerintelor (2)

- **Analiza cerințelor :**

- rezolvarea posibilelor conflicte între diferite cerințe utilizator,
- prioritizarea cerințelor,
- rafinarea cerințelor utilizator prin exprimarea lor într-o manieră mai formală

- Rezultatul analizei cerințelor:

- **specificat într-un document** numit, de regula

Documentul Cerințelor Software (Software Requirements Document) – **SRD**

- adesea, SRD conține și
 - cerințele utilizator – caz în care nu mai este necesar URD
 - cerințe de sistem (hardware)

➤ **Documentul de specificare a cerințelor (SRD):**

- se folosește pentru estimarea costurilor dezvoltării și pentru planificarea dezvoltării;
- este documentul de referință pentru testarea de sistem și testarea de acceptare

Proiectarea (1)

- **Proiectarea arhitecturală (numită și proiectarea de sistem)**
 - Se stabilește arhitectura hardware-software a sistemului care va implementa cerințele: se definesc **subsistemele și interfețele lor**, se decide asupra **tehnologiilor software care vor fi folosite în implementare și asupra componentelor hardware ale sistemului**.
 - Toate cerințele specificate trebuie să fie acoperite de arhitectura sistemului.
 - Se alege solutia optima dintre alternativele posibile de implementare a cerintelor.
 - Rezultat: **ADD** (Architectural Design Document); conține: descrierea subsistemelor și a interfețelor lor, repartitia artefactelor software la momentul executiei pe echipamentele sistemului.
 - Diferitele subsisteme definite în ADD sunt implementate de echipe diferite:
 - ADD se folosește pentru specificarea testelor de integrare.

Proiectarea (2)

•*Proiectarea de detaliu*

- Se descompun subsistemele pana la nivel de module (unități program) care se implementeaza in limbajul de programare ales (functii, clase).
- Rezultat: **DDD** (Detailed Design Document); contine: descrierea interfetei si a rolului fiecarui modul, specificatia functionala a fiecarui modul, comunicatia intre module, procesele din sistem și sincronizarea lor, s.a.
- DDD se foloseste pentru definirea testelor unitare

Implementarea modulelor

Implementarea cuprinde codificarea si testarea separata a modulelor definite in etapa de proiectare de detaliu.

- Este cea mai bine stapanita si cea mai bine “utilata” dintre toate activitatile de dezvoltare software. Exista:
 - Limbaje de programare, compilatoare, instrumente de localizare a surselor de erori (debug-ere) – integrate in medii de dezvoltare
 - Instrumente software pentru testarea automata
- Reprezinta circa 15-20% din efortul total de dezvoltare a unui produs program.

Integrarea si testarea

Integrarea

- Modulele care au fost codificate și testate independent sunt integrate treptat in subsisteme, pana la nivel de sistem.
- Se verifica interactiunea si comunicarea intre componentele integrate prin **teste de integrare**.
- Rezultatul integrarii: o versiune executabila a produsului software

Testarea de sistem

- Se verifica daca produsul obtinut (sistemul hard-soft) **satisface toate cerintele specificate**
- Efectuata in interiorul companiei care dezvoltă produsul software.

Testarea de acceptare

- **Se verifica impreuna cu clientul** daca sistemul **satisface cerintele utilizator specificate**
- Produsul software este instalat si testat in mediul in care va opera
- Efectuata de o echipa de testare independenta care include clientul si viitori utilizatori
- Testare alfa/beta; testarea alfa: testarea efectuata impreuna cu clientul.
- Testarea beta: produsul este distribuit unor utilizatori selectati, care vor raporta eventualele caderi produse in timpul utilizarii.

Operarea si Intretinerea

- **Operarea:** - utilizarea efectiva a software-ului in mediul real de functionare
 - începe dupa livrarea software-ului la client
- **In timpul operarii pot apărea necesități de modificare:**
 - Corectie defecte (bug-uri) descoperite in timpul operarii
 - Imbunatatire functii existente
 - Adaptare la noi tehnologii
 - Adaugare noi functii
- **Intreținerea (mentenanța): efectuarea modificarilor necesare în timpul operarii.**
 - poate fi efectuata de catre echipe / persoane diferite de cele care au participat la dezvoltarea produsului software;
 - poate necesita efectuarea oricarei activitati tehnice din perioada dezvoltarii;
 - **costul întreținerii depinde de calitatea proiectarii, a documentarii și testarii.**

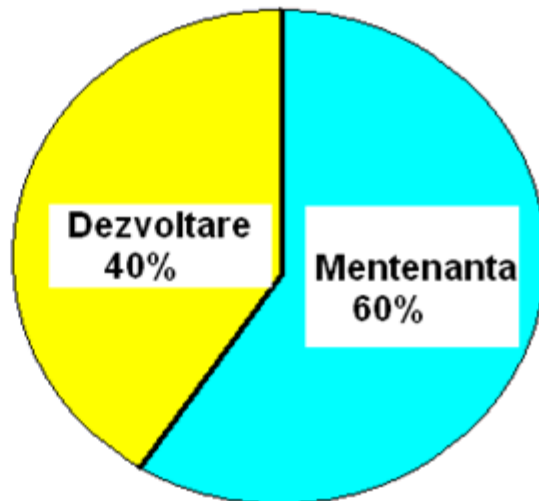
Distributia efortului pe parcursul vieții unui produs software

Viața unui produs software = perioada de dezvoltare + perioada de operare

Dezvoltare



Dezvoltare/Mentenananta



Asigurarea calitatii

- **Scop: asigurarea cerintelor tehnice si a standardelor de calitate pentru**
 - procesul de dezvoltare
 - produsul final
- **Activitati:**
 - Alegerea modelului de dezvoltare
 - Alegerea metodelor si a standardelor de specificare, proiectare si implementare
 - Revizii, pe tot parcursul procesului de dezvoltare
 - Definirea strategiilor de testare
 - Definirea metodelor de documentare
 - Definirea metricilor de evaluare a artefactelor procesului de dezvoltare, definirea instrumentelor de masurare

Managementul proiectului (1)

Activitati:

- Scrierea propunerii pentru obtinerea proiectului de dezvoltare a produsului software
- Estimari asupra necesarului de resurse umane pentru dezvoltarea unui produs software
- Decizii asupra (re-) alocarilor de resurse umane pe activitati si etape ale dezvoltarii
- Stabilirea si planificarea etapelor de dezvoltare
- Revizii ale documentelor elaborate pe parcursul dezvoltarii
- Selectia si evaluarea personalului
- Scrierea si prezentarea de rapoarte pe parcursul procesului de dezvoltare
- Studiul de fezabilitate

Managementul proiectului (2)

Studiul de fezabilitate

➤ Trebuie sa preceada initierea proiectului software

- Se determina obiectivele proiectului
- Se analizeaza procesul care ar trebui imbunatatit/ produsul pe care ar trebui sa-l inlocuiasca noul produs software
- Se estimeaza costurile

➤ Pe baza studiului de fezabilitate se poate decide că:

- Proiectul poate sa inceapă (este realizabil)
- Proiectul se abandoneaza:
 - Realizarea produsului software nu aduce beneficii clientului (*Return on Investment* - *ROI*)
 - Costurile de dezvoltare sunt mai mari decat finantarea oferita de client
 - Realizarea nu este posibila cu echipamentele pe care le are/vrea clientul
 - Etc.

Riscurile unui proiect software

De ce poate eșua un proiect software?

- **Factori de experienta:**
 - a managerului
 - a echipei
 - a organizatiei in care se dezvolta produsul
- **Factori de planificare:**
 - estimarea necesarului de resurse umane
 - estimarea perioadelor de timp pentru diferite activitati
 - definirea responsabilitatilor
- **Factori tehnologici:**
 - noutatea tehnologica (tehnologii inca nevalidate in practica)
 - metodele de dezvoltare
 - instrumentele de dezvoltare
- **Factori externi:**
 - calitatea specificatiei cerintelor: specificatia cerintelor incompleta, inconsistenta, ambigua
 - stabilitatea cerintelor (cerintele se modifica pe parcursul dezvoltarii)
 - stabilitatea si disponibilitatea altor factori de care depinde proiectul: mediul de utilizare, intarzieri in achizitionarea unor echipamente necesare in dezvoltare, s.a.