Ingineria Programării

Curs 1 – 17 Februarie 2014

Adrian Iftene
adiftene@info.uaic.ro

Cuprins

- Context
- Motivație
- Erori celebre
- Statistici
- Definiții
- Etapele dezvoltării programelor
- Modele de dezvoltare
  - Cascadă
  - Spirală
  - Prototip
  - RUP

Context

- **Aplicații de dimensiuni mari** (milioane de linii de cod) scrise pe durata a câteva luni, ani,…
- **Echipe de lucru**: Project manageri, Analiști, Arhitecți, Programatori, Testeri, Ingineri de suport (de ordinul 10, 100, 1.000 de persoane,…)
- Soluția acestor probleme este o soluție scrisă într-un limbaj de **programare orientat-obiect**

Motivație 1

- Din ce în ce mai multe sisteme sunt controlate de software: controlul traficului (aerian, feroviar, auto, naval, etc.), băncile, telefonia mobilă, Internet
- Economiile tuturor statelor depind de software
- Ingineria programării propune teorii, metodologii și instrumente pentru dezvoltarea de software profesional

Motivație 2

- **1946** Goldstine și von Neumann: "**1.000 de instrucțiuni** reprezintă o limită superioară rezonabilă pentru complexitatea problemelor ce pot fi concepute ca rezolvabile cu ajutorul calculatorului"

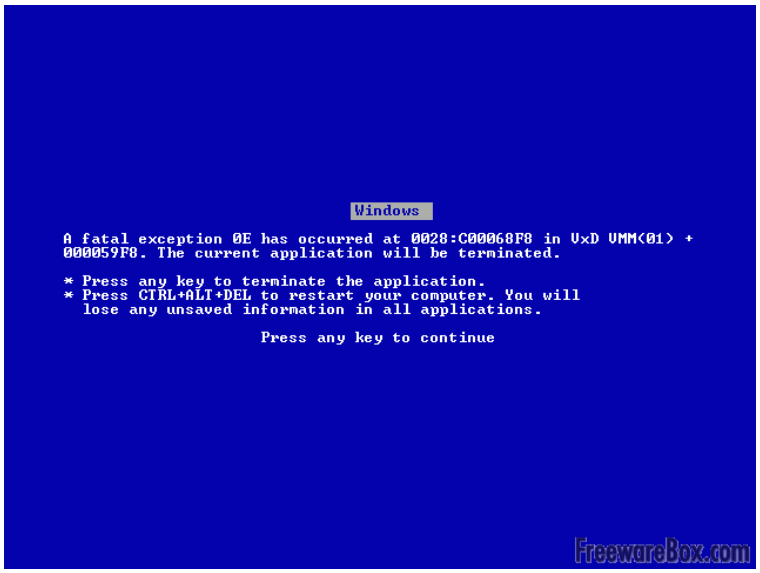
- Sistemul de rezervare a biletelor pentru compania aerian a KLM conținea, în anul **1992**, **două milioane de linii** de cod în limbaj de asamblare

Motivație 3

- Sistemul de operare System V versiunea 4.0 (UNIX) a fost obținut prin compilarea a **3.700.000 linii de cod**
- Programele scrise pentru naveta spațială NASA au circa **40 de milioane de linii de cod** obiect
- Pentru realizarea sistemului de operare IBM OS360 au fost necesari **5.000 de ani-om**

Motivație – Erori celebre 1

- Facturi imense la energia electrică pentru pensionari, recalcularea pensiilor



- IBM OS360 conținea la fiecare relansare 1.000 de greșeli. *Resemnare...*

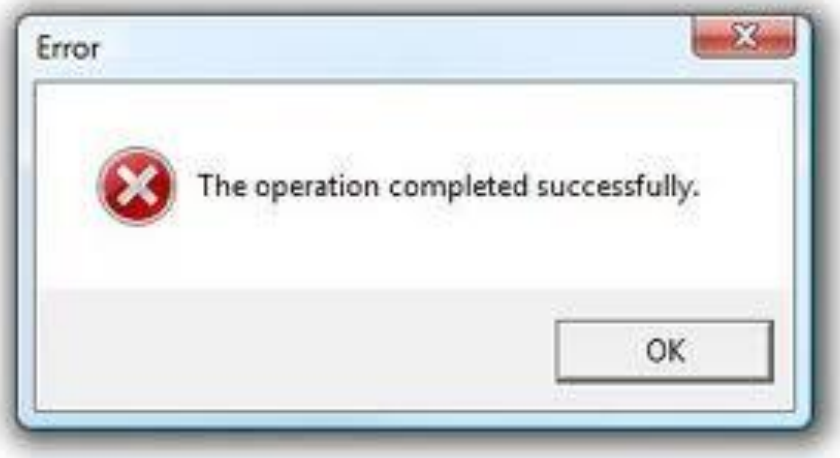
Motivație – Erori celebre 2

- *Programatorul ghinionist de la o bancă*: Banca dorea să trimită la clienți prin poștă o scrisoare care să le semnaleze diverse servicii
- Programatorul a scris un program ce selecta 2.000 de clienți și le scria o scrisoare personalizată
- În procesul de testare acesta a folosit un nume fictiv de client **Rich Bastard**
- Din păcate 2.000 de clienți au primit o scrisoare care începea "Dear **Rich Bastard**, ..."

Motivație – Erori celebre 3

- Sacramento: un dentist primește în căsuța poștală într-o săptămână 16.000 de formulare pentru plata taxelor – "*It was a computer problem*" a declarat un oficial

- "*Failure to convert English measures to metric values was the root cause of the loss of the Mars Climate Orbite…*"



Motivație – Erori celebre 4

- Pierdere **vehicul** explorare Venus. *Ah, era de fapt .,. in FOR!...*
- Sistem de **avertizare anti-rachetă activat.** *Atacăm sau nu?*
- Ariane 5 explodeaza. *Cost articii:* ***500.000.000$***

Motivație – Statistici 1

- Acestea nu sunt doar “erori amuzante”

- Proiectele mari software pot fi cele mai complicate produse realizate de cineva
- Să ne uităm pe **statistici**
  - Vom considera că un proiect software are **succes**, dacă este realizat într-un timp rezonabil și cu un buget rezonabil
  - Un **eșec** al unui produs software are loc atunci când produsul nu este realizat sau când nu poate fi folosit

Motivație – Statistici 2

- **Studii de succes: USA'82 – Gibson & Singer**
- 18 proiecte
  - Succes: 17%
  - Parțial în folosință: 28%
  - Satisfacătoare 11%
  - Eșec: 22%
  - Neevaluate: 11%
- Motivele eșecurilor
  - Probleme de organizare
  - Noile metode de lucru și politicile salariale
  - Modificările neprevăzute în afacere

Motivație – Statistici 3

- **Studii de succes – ONNI'88 (Finland)**
- Din peste 100 proiecte
  - Succes: 33%
  - Cu probleme: 42%
  - Eșec: 25%
- Motivele eșecurilor
  - Slaba pregătire a inginerilor software
  - Resurse insuficiente
  - Probleme de management

Motivație – Statistici 4

- **The Robbins-Gioia Survey (2001),** Alexandria – Virginia, made a study over the perception by enterprises of their implementation of an E.R.P. (Enterprise Resource Planning) package.
- 232 survey respondents (36 % had, or were in the process of, implementing an ERP system)
- **51 % viewed their ERP implementation as unsuccessful**,
- **46 % did not understood how to use the system**
- 56 % of survey respondents noted their organization has a program management office (PMO) in place, and of these respondents, only 36 % felt their ERP implementation was unsuccessful

Motivație – Statistici 5

- **The Conference Board Survey (2001)**
- At 117 companies that attempted ERP implementations
- **34 % were very "satisfied"**
- **58 % were "somewhat satisfied"**
- **8 % were unhappy** with what they got.
- **40 % of the projects failed** to achieve their business case within one year of going live

- The companies that did achieve benefits said that achievement took **six months longer** than expected.
- Implementation costs were found to average 25 % over budget

IP – Definiție 1

- Prima definiție a ingineriei programării (NATO,1968): *Ingineria programării este stabilirea și utilizarea de principii inginerești solide pentru a obține în mod economic programe care sunt sigure și funcționează eficient pe mașini de calcul concrete*
- O definiție mai recentă (și mai rezervată, *IEEE Standard Glossary of Software Engineering Tehnology, 1983): Ingineria programării reprezintă abordarea sistematică a dezvoltării, funcționării, întreținerii, și retragerii din funcțiune a programelor*

IP – Definiție 2

- Se referă la două lucruri:
  - "software engineer" care a înlocuit termenul de "programmer”
  - "Software Engineering" care este folosit pentru a descrie "building of software systems which are so large or so complex that they are built by a team or teams of engineers“ – *Fundamentals of Software Engineering* (Ghezzi, Jazayeri, and Mandrioli)

IP – Definiții 3

- **FreeDictionary**: “The process of **manufacturing** software systems. A software system consists of executable computer code and the supporting documents needed to manufacture, use, and **maintain** the code.”
- **Webopedia.com**: “The computer science discipline concerned with **developing large applications**. Software engineering covers not only the technical aspects of building software systems, but also **management issues**, such as **directing programming teams**, **scheduling**, and **budgeting**. “

IP – Definiții 3

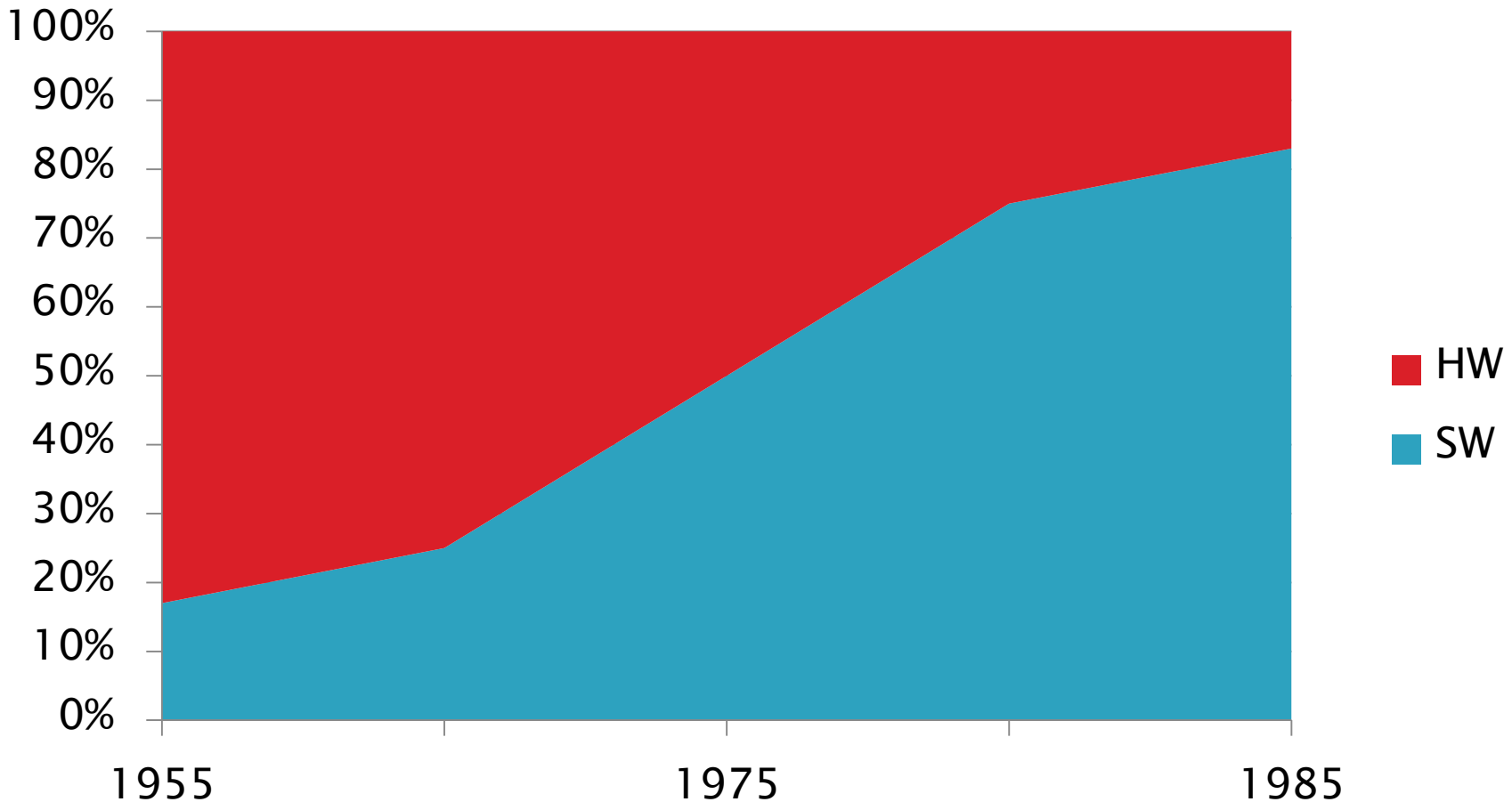
- **Wikipedia**: "**Software engineering** is the application of a systematic, disciplined, quantifiable approach to the **development**, operation, and **maintenance** of software, and the study of these approaches"

- **Answers.com**: "The **systematic** application of scientific and technological knowledge, through the medium of sound engineering principles, to the production of computer programs, and to the **requirements definition**, **functional specification**, **design description**, **program implementation**, and **test methods** that lead up to this code"

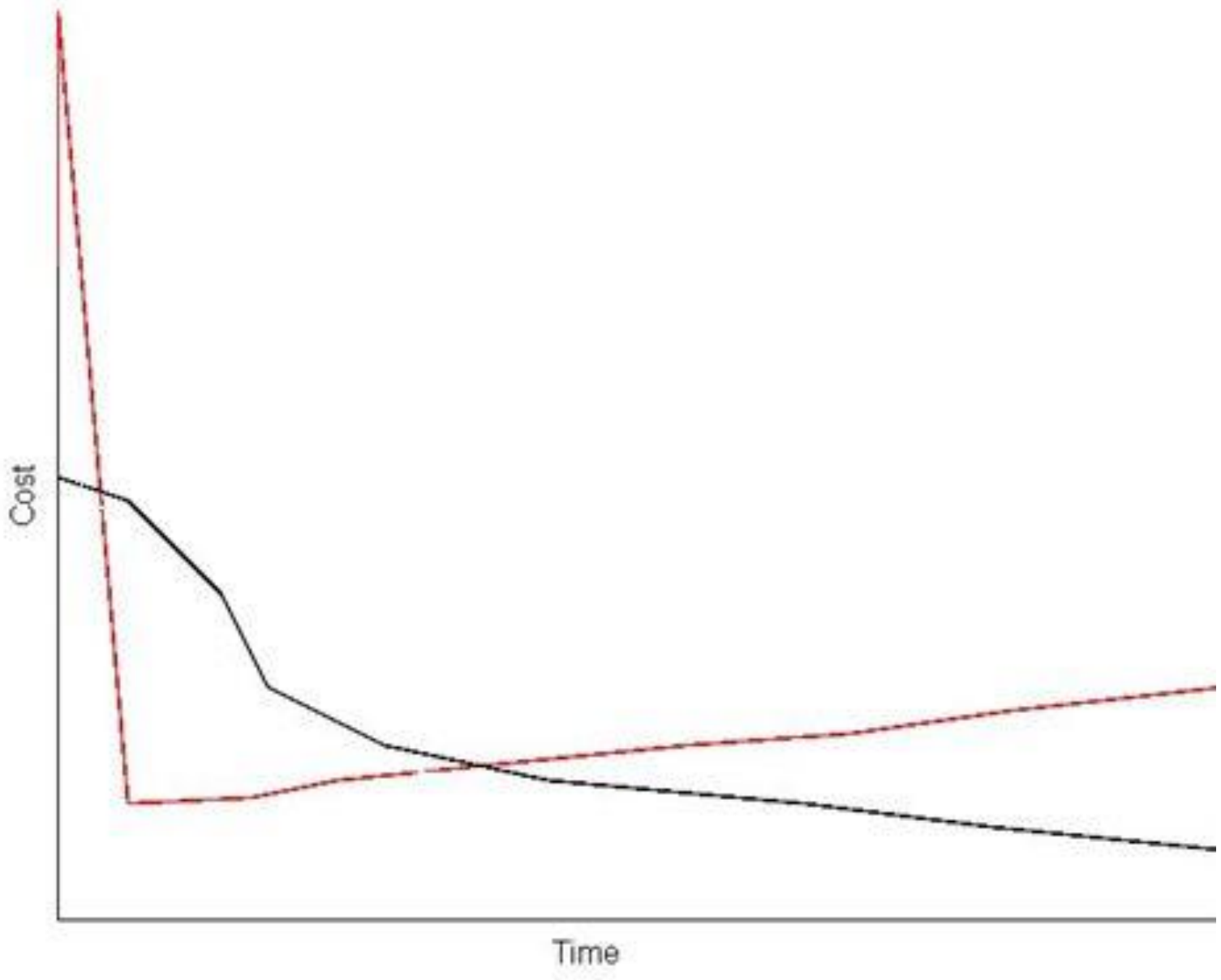
IP – Pe scurt

- Este o disciplină inginerească care se ocupă de toate aspectele dezvoltării unui program de dimensiuni mari, de către o echipă de dezvoltatori
- Propune adoptarea unei abordări sistematice și organizate a procesului de dezvoltare software
- Propune folosirea tehnicilor și instrumentelor adecvate având în vedere
  - problema care trebuie rezolvată
  - restricțiile impuse
  - resursele disponibile

Costurile produselor software



Software build (black) vs. buy (red)



Costurile produselor software

- În prezent costurile acestora sunt mai mari decât costurile rezervate componentelor hardware
- Costul întreținerii unui program este de regulă mai mare decât costul realizării unui program
- **Software** – reprezintă programele și documentația aferentă acestora

Atributele unui program bun

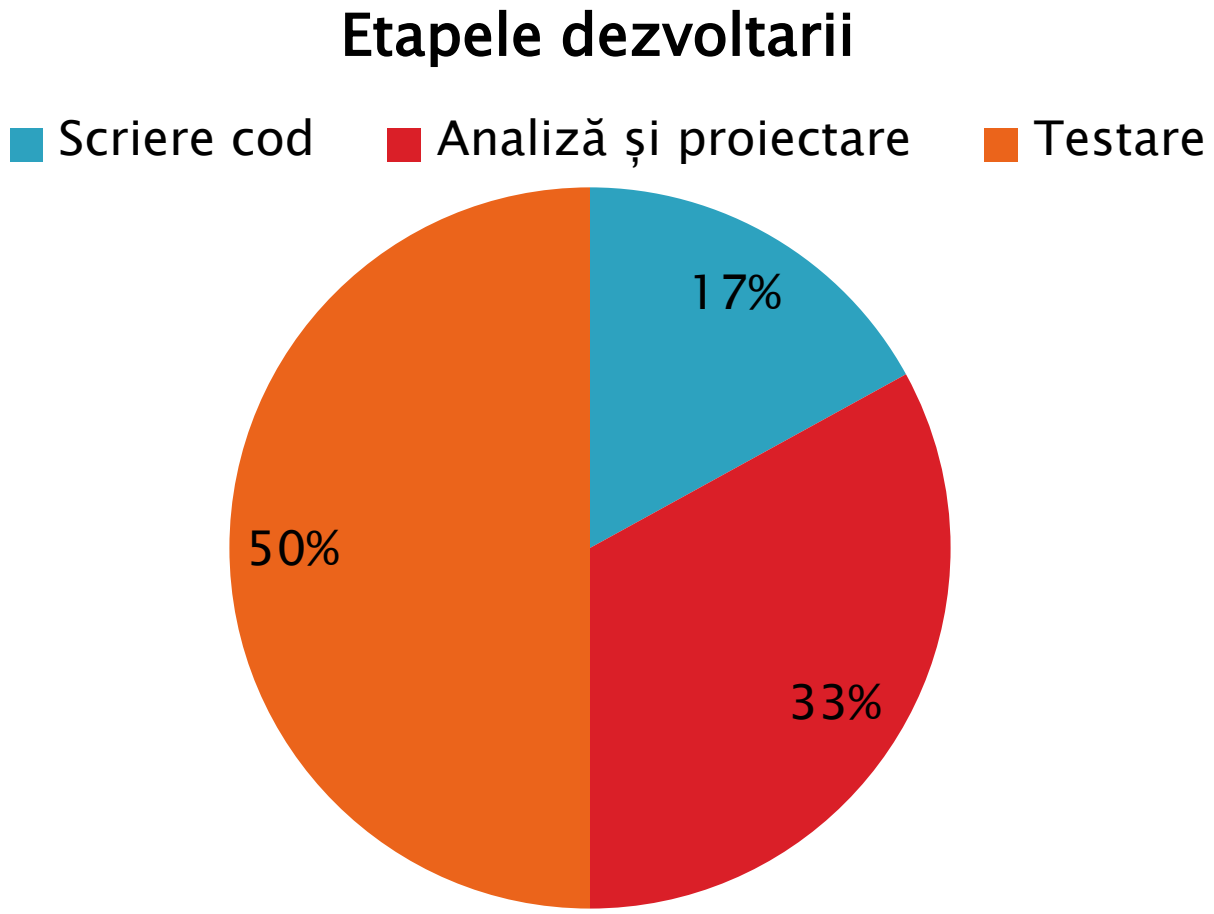
- Să ofere funcționalitățile cerute
- Să fie ușor de modificat, completat, schimbat
- Să fie sigur
- Să nu irosească resurse hardware
- Să fie ușor de folosit

Încadrarea IP

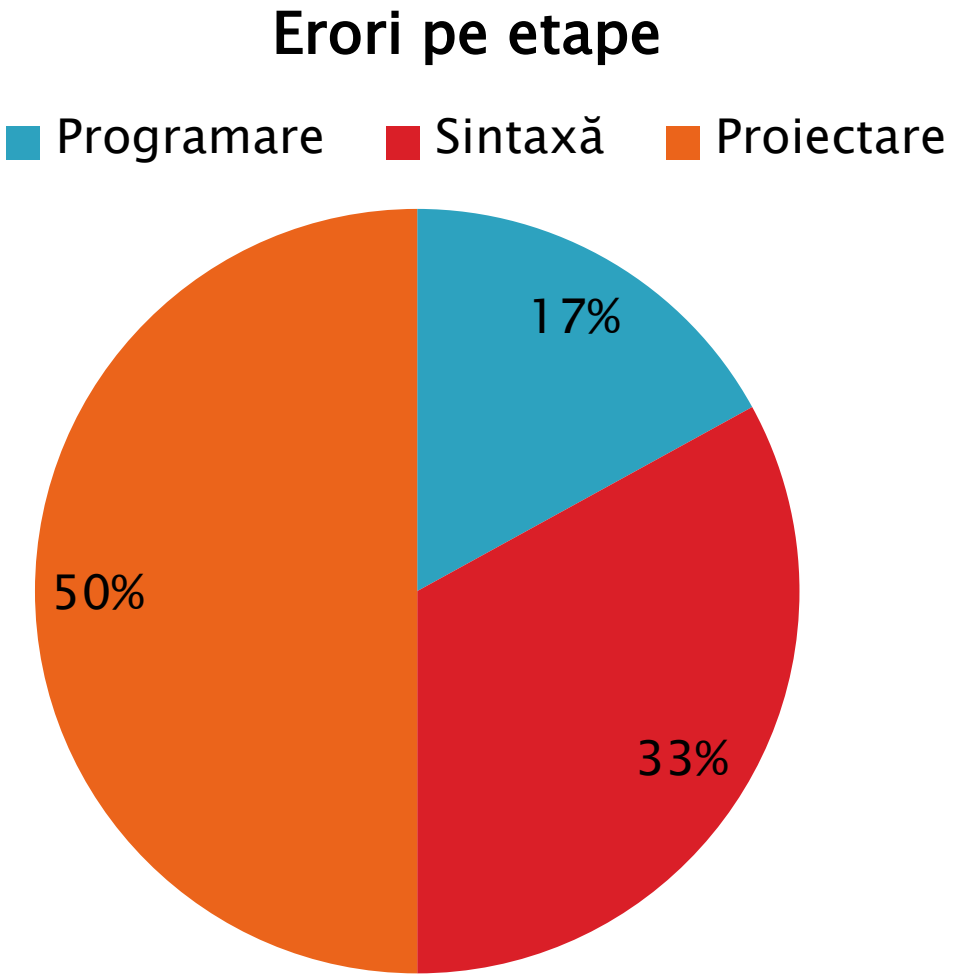
- **IP vs informatică**
  - Informatica se ocupă de **aspectele teoretice** ale dezvoltării software
  - IP se ocupa de **aspectele practice** ale dezvoltării software

- **IP vs ingineria sistemelor**
  - Ingineria sistemelor se ocupa de **toate aspectele dezvoltării** sistemelor de calcul (hardware, software, ingineria proceselor)
  - IP este o parte din ingineria sistemelor și se ocupă de:
    - Specificarea cerințelor
    - Proiectarea arhitecturală
    - Implementare
    - Testare
    - Integrare
    - Deployment

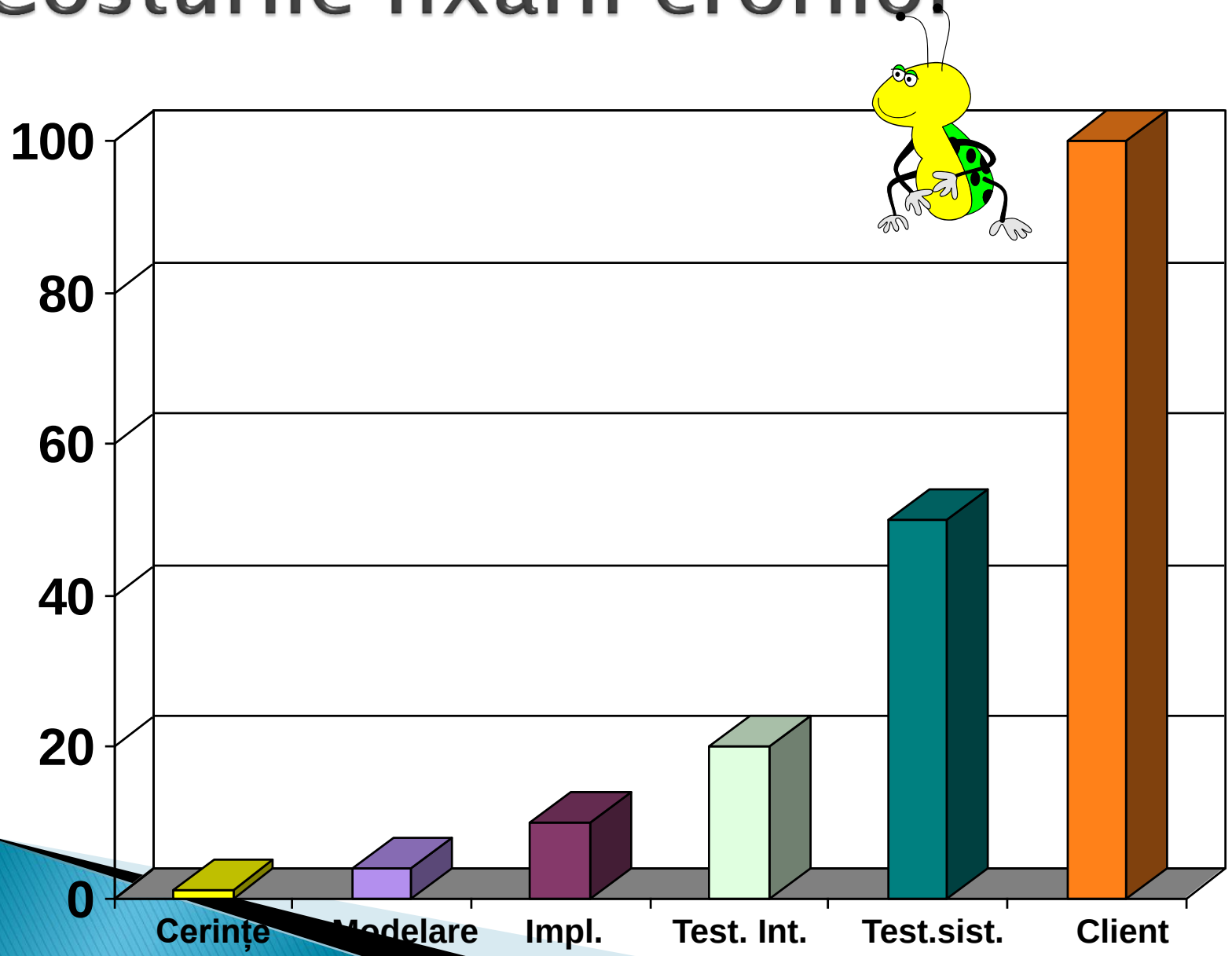
Repartizarea costurilor pe etape



Repartizarea erorilor pe etape



Costurile fixării erorilor



Dificultăți în IP

- Sistemele mai vechi care trebuie întreținute și actualizate
- Eterogenitatea sistemelor software/hardware
- Presiunea pentru a livra programul repede, cu un cost mic și cu o calitate bună

Modele de dezvoltare

- Pentru a dezvolta un program este nevoie de:
  - O înțelegere clară a ceea ce se cere
  - Un set de metode și instrumente de lucru
  - Un plan de acțiune

- Plan de acțiune = șablon = model de dezvoltare

Etapele dezvoltării programelor

- Analiza cerințelor (Requirements analisys)
- Proiectarea arhitecturală (Arhitectural design)
- Proiectarea detaliată (Detailed design)
- Scrierea codului (Implementation)
- Integrarea componentelor (Integration)
- Validare (Validation)
- Verificare (Verification)
- Întreținere (Maintenance)

Analiza cerințelor

- Se stabilește *ce anume vrea clientul ca programul să facă*
- Scopul este înregistrarea cerințelor într-o manieră cât mai clară și mai fidelă
- Probleme
  - Comunicare
  - Negociere
  - Sfătuirea clientului

Proiectarea

- Arhitecturală
  - Din motive de complexitate, programele mari nu pot fi concepute și implementate ca o singură bucată
  - Programul este împărțit în module sau componente mai simple, care pot fi abordate individual

- Detaliată
  - Se proiectează fiecare modul al aplicației, în cele mai mici detalii

Implementare + integrare

- Implementare
  - Proiectul detaliat este transpus într-un limbaj de programare
  - Acesta se realizează modular, pe structura rezultată la proiectarea arhitecturală
- Integrare
  - Modelul big-bang
  - Modelul incremental

Validare și verificare

- **Validare:** ne asigurăm că programul îndeplinește cerințele utilizatorului
  - *Construim produsul corect?*

- **Verificare:** ne asigurăm că programul este stabil și că funcționează corect din punctul de vedere al dezvoltatorilor.
  - *Construim corect produsul?*

Întreținere

- După livrare
  - Sunt descoperite greșeli ce trebuie reparate
  - Pot apărea schimbări în specificații
  - Pot apărea noi cerințe
  - Instruirea celor ce vor folosi produsul

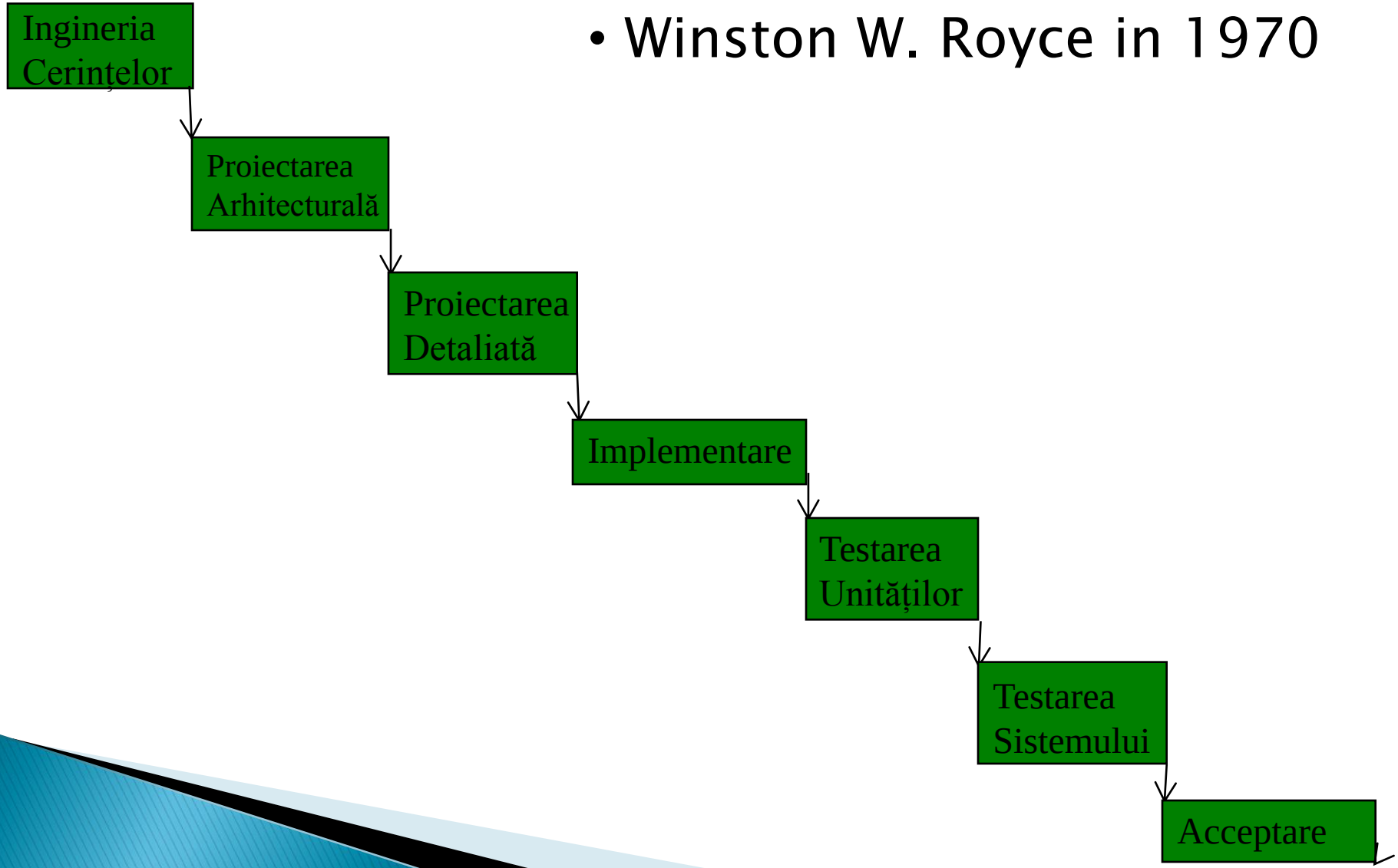
- Întreținere = gestionarea acestor tipuri de probleme

Modele de dezvoltare

- *Cum* efectuăm activitățile indicate de etapele dezvoltării programelor
- Exemple de modele de dezvoltare:
  - Ad-hoc: descurcă-te cum poți
  - Modelul în cascadă (cu feedback)
  - Prototipizare
  - Modelul în spirală
  - RUP (Rational Unified Process)
  - XP (Extreme Programming)
  - Agile
  - Lean, Scrum
  - MDD, AMDD
  - TDD

Modelul în Cascadă (Waterfall)

- Winston W. Royce in 1970

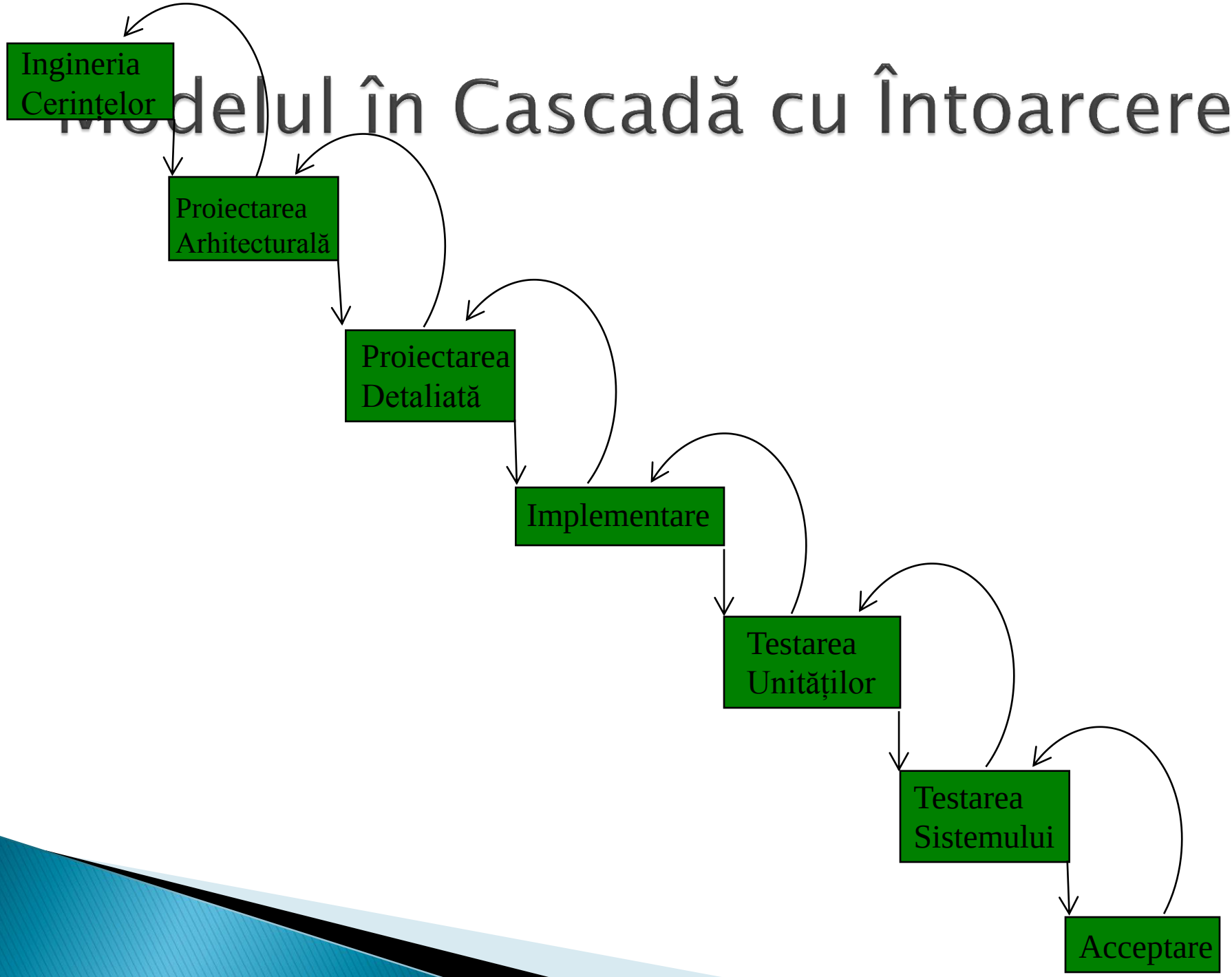


Modelul în Cascadă +-

- +: Împarte o sarcină complexă în pași mai mici
- +: Ușor de administrat și controlat
- +: Fiecare pas are ca rezultat un produs bine definit
- +: Tot timpul știm unde ce am făcut până în acel moment și știm ce mai avem de făcut

- -: Erorile se propagă între pași
- -: Nu exista mecanisme de reparare a erorilor

Modelul în Cascadă cu Întoarcere



Modelul în cascadă cu întoarcere +-

- +: Oferă cadrul pentru remedierea erorilor din pasul precedent

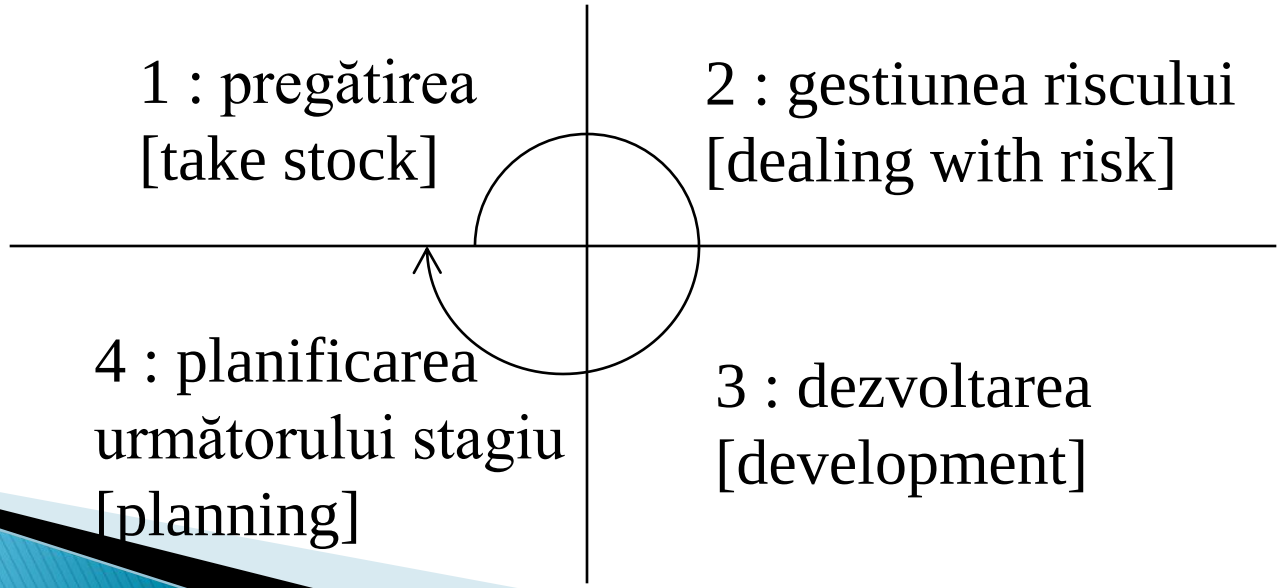
- -: Erorile la pasul i care sunt descoperite la pasul i + 2 nu sunt remediate

- -: Clientul vede produsul final abia la sfârșitul dezvoltării

Modelul în Spirală (Spiral)

- Studiul de fezabilitate
- Analiza cerințelor
- Proiectarea arhitecturii
- Implementarea

Pentru fiecare pas, se fac următoarele activități:



Modelul în Spirală +-

- +: Păstrează avantajele modelului în cascadă
- +: la în calcul noțiunea de risc

- Exemple de riscuri:
  - O firmă concurentă lansează un produs similar
  - Un arhitect părăsește echipa
  - Clientul schimbă cerințele
  - O echipă nu respectă termenele de livrare

Prototipizare (Prototyping)

- Tipuri de prototipuri
  - **De aruncat** (throw-away)
    - Scop: clarificarea specificațiilor
    - Se dezvoltă repede, orice altceva e secundar (quick-and-dirty)
    - Util în a rezolva “architecural/technology spikes”
    - Programul “adevărat” este scris apoi de la 0
  - **Evoluționar**
    - Scop: construire incrementală a produsului final
    - Se construiește un nucleu funcțional la care se adaugă apoi noi funcționalități

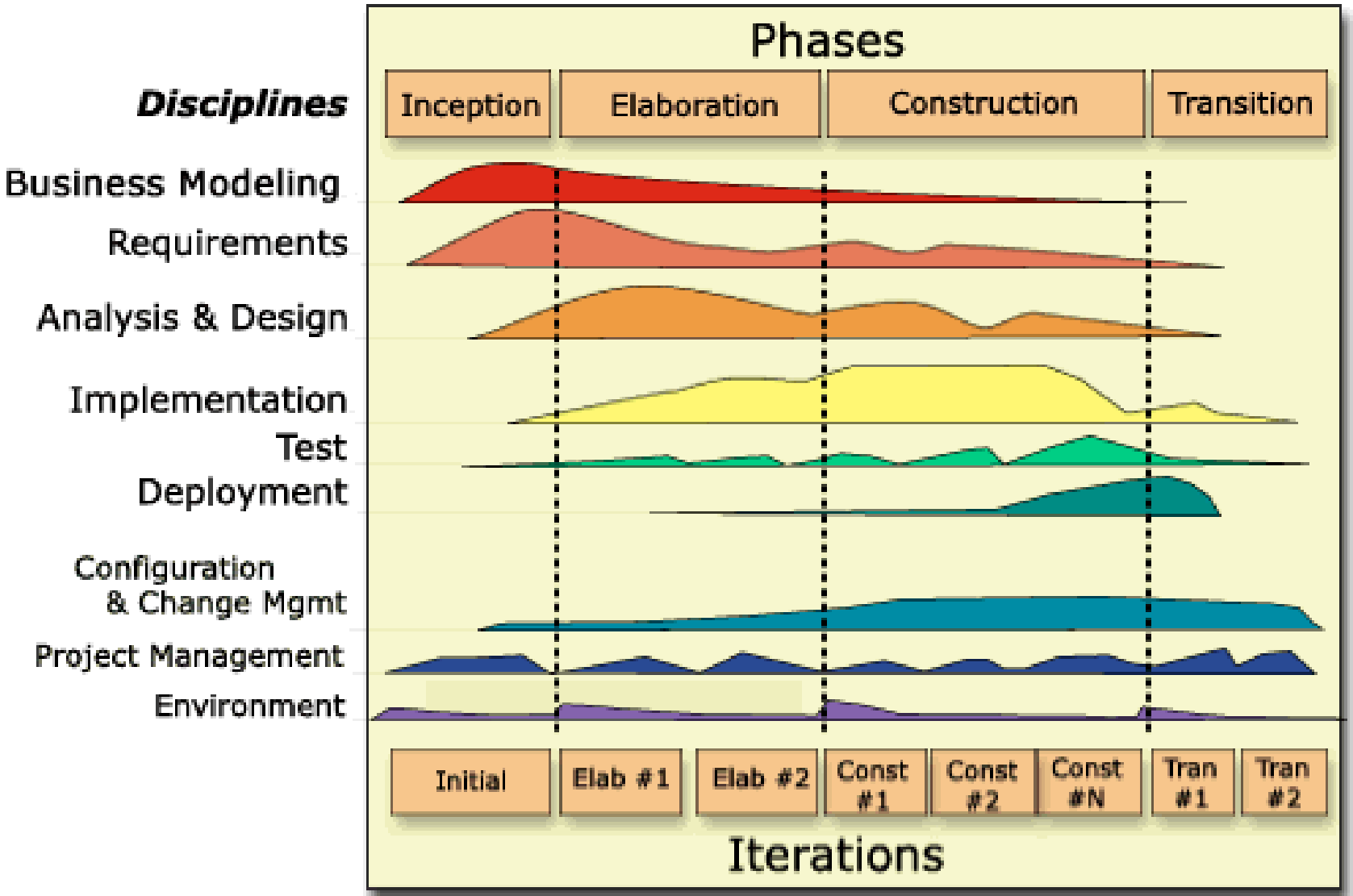
Prototipizare +-

- +: Se poate elimina lipsa de claritate a specificațiilor
- +: Clienții pot schimba cerințele (e ieftin de gestionat)
- +: Întreținere ieftină (verificare pe parcurs)
- +: Se poate facilita instruirea utilizatorilor

- -: Mediu artificial, probleme ascunse
- -: *Da' nu-i aproape gata?! De ce mai durează atât?*
- -: *Putem să schimbăm specificațiile? Pai aș vrea și...*
- -: *Adică munca mea este aruncată la gunoi?*

RUP (Rational Unified Process)

- Model iterativ folosit de IBM din 2003



RUP

- Ingineria funcționalității. Sunt sintetizate necesitățile funcționale.
- Are la baza 4 etape:
  - **Inception**: pentru validarea costurilor și bugetului, studiu de risc, înțelegerea cerințelor
  - **Elaboration**: analiza domeniului problemei, arhitectura proiectului este stabilită
  - **Construction**: construcția sistemului, se obține prima versiune a sistemului
  - **Transition**: tranziția la sistemul din producție

Concluzii

- Statisticile ne demonstrează importanța folosirii unor tehnici inginerești în dezvoltarea de software
- Definițiile IP-ului folosesc cuvinte cheie precum: *metode inginerești*, *proiecte de dimensiuni mari care sunt implementate în echipă*, *programe sigure care funcționează eficient*, *dezvoltare, întreținere, planificare, buget*
- Etapele necesare dezvoltării proiectelor de dimensiuni mari
- Modele de dezvoltare: Ad-Hoc, Cascadă, Spirală, Prototip, RUP

Bibliografie

- Ovidiu Gheorghieș: http://www.infoiasi.ro/~ogh/files/ip/curs-01.pdf
- Andy Kramek, New Software – Build or Buy? A Personal View: http://weblogs.foxite.com/andykramek /archive/2009/07/25/8674.aspx

Links

- Internet
- Wikipedia
- Failure rate: http://www.it-cortex.com/Stat_Failure_Rate.htm
- RUP in the dialogue with Scrum: http://www.ibm.com/developerworks/rational/library/feb05/krebs/