Universitatea din Pitești
Facultatea de Electronică și Calculatoare

Ingineria Programelor

Lect. univ. dr. Adrian Țurcanu

05.10.2016

Cuprins

Obiective și desfășurare

- Familiarizarea cu conceptele fundamentale ale ingineriei software.
- Prezentarea comparativă a unor metode și modele de dezvoltare a sistemelor software.
- Prezentarea rolului testării și verificării formale în programare.

- Curs: 2 ore/săptămână

- Laborator:2 ore/săptămână
- Proiect:1 oră/săptămână

Evaluare

- Prezența: 10%
- Activitatea la laborator: 10%
- Proiect: 30%
- Examen scris: 50%

Cuprinsul cursului

- Introducere în ingineria programării
- Procese software: etape, modele
- UML: cazuri de utilizare, diagrame de interacțiune, diagrame de clase, diagrame stări, etc.
- Metode de testare a programelor
- Verificare și validare software
- Studiu de caz: Modelarea, testarea și verificarea unor sisteme

Bibliografie minimală

1. F. Ipate, Modelare orientată pe obiecte cu UML, Editura Universității Pitești, 2001.
2. I. Sommerville, Software Engineering, 6th Edition, Addison Wesley, 2001.
3. M. Roper, Software Testing, McGraw-Hill, 1994.
4. G. Varvara, Ingineria Programării - curs, Ed. ConsPress, 2013.
5. M. Holcombe, F. Ipate, Correct Systems: Building a Business Process Solution, Springer Verlag, 1998.

Istoric

- Primele calculatoare digitale nu făceau o distincție clară între hardware și software, presupuneau progrmarea folosind codul mașină și erau operate, în general, de ingineri.
- Distinția între hardware și software apare odată cu arhitectura Von Neumann și primele limbaje de programare de nivel înalt.
- Termenul de "software engineering" (inginerie software, ingineria programelor) a fost utilizat pentru prima dată în anul 1968 la o conferință organizată de NATO în Germania și a apărut în contextul unei "crize software" determinată de limitările tehnologiilor disponibile la vremea respectivă, în domeniul dezvoltării programelor.

- Încă din acea perioadă au devenit clare o serie de aspecte legate de dezvoltarea programelor cum ar fi:
  - programele de mari dimensiuni ridică mult mai multe probleme decât cele de mici dimensiuni. Metodele de dezvoltare existente la momentul respectiv erau inadecvate dezvoltării de programe mari;
  - componentele hardware nu reprezintă factorul cel mai important;
  - un program nu este o entitate statica, ci el evoluează în timp;
  - un program trebuie să fie ușor de înțeles și adaptat.

- Ingineria programării a apărut ca o disciplină care să furnizeze cadrul pentru construirea de software.
- Ingineria programelor consideră programul ca un obiect complex și are ca scop definirea de tehnici de "fabricație" justificate de teorie sau de practică.
- Această disciplină își propune să combine arta programării cu rigoarea stiințelor inginerești pentru a putea livra programe la timp și în mod economic.
- Ingineria programelor s-a dezoltat odată cu creșterea accentuată a complexității programelor (de exemplu, sistemul de rezervare a biletelor pentru compania aeriană KLM conținea, în anul 1992, aproximativ două milioane de linii de cod în limbaj de asamblare).

Definiții

In standardul IEEE 1993, ingineria software este definită astfel: "Aplicarea unei abordari sistematice, disciplinate și măsurabile în dezvoltarea, operarea și întreținerea software-ului, adică aplicarea ingineriei pentru software. De asemenea, studiul unor asemenea abordări."

Ingineria programelor (software engineering) este disciplina care se ocupă cu toate aspectele producției de software de la primele faze ale specificației până la mentenanța acestuia după ce a fost furnizat clientului pentru utilizare.

Ce înseamnă software?

- programe pentru calculator
- documentația asociată acestor programe
- datele referitoare la configurația necesară pentru funcționare

Caracteristici ale software-ului

- Spre deosebire de hardware, software-ul nu este fabricat ci dezvoltat.
- Costurile dezvoltării de software sunt concentrate pe inginerie(concepție).
- Software-ul nu-și pierde calitățile în timp așa cum se întâmplă cu hardware-ul.
- Majoritatea produselor software sunt construite pentru a raspunde anumitor cerințe.

Ce reprezintă un sistem software?

- un număr de programe separate
- fișiere de configurație necesare pentru funcționare
- documentație care descrie structura sistemului
- documentatie pentru utilizator

Tipuri de sisteme software

- generice: produse de firme de dezvoltare și vândute pe întreaga piață
- personalizate(pentru client): contractate și dezvoltate pentru un anumit client

Principala diferență între cele două tipuri este aceea că, în timp ce, în cazul sistemelor generice specificația este controlată și dezvoltată de catre firma care dezvolta sistemul, în cazul sistemelor personalizate specificația este dezvoltată și controlată de către client.

Ce înseamnă software de calitate?

- satisface cerințele utilizatorului
- nu conține erori
- este flexibil: permite schimbări făcute după livrare, corectări de erori, schimbări de funcționalitate
- are un preț rezonabil atât pentru dezoltare cât și pentru mentenanță
- realizează funcționalitatea dorită.

Cum se construiește un sistem software de calitate?

- Se folosește un proces cu faze clare, fiecare având un produs final (document, program, etc.).
- Cerințele sunt identificate cât mai repede.
- Problema este analizată și împărțită în subprobleme.
- Verificarea si validarea prin testare sau chiar demonstrații sunt esențiale.
- Se folosesc metode, modele și limbaje relevante.
- Se folosesc utilitare adecvate.

Complexitatea sistemelor software

- complexitatea problemei - este determinată de cerințele neclare sau care se modifică ale clientului, de limitele de performanță, timp și bani, de specificații complexe sau ambigue.
- dificultatea de a administra procesul de dezvoltare - un volum mare de muncă determină folosirea unei echipe mai mari ceea ce atrage potențiale probleme de comunicare și coordonare.
- flexibilitatea sistemului - sistemul trebuie dezvoltat de la abstract la concret.
- problemele de caracterizare a comportării sistemelor discrete - mulțimea stărilor unui sistem discret de mari dimensiuni este foarte mare, iar acestea nu pot fi caracterizate unitar.

Probleme cu sistemele de mare dimensiune

- realizarea acestora durează, în medie cu 50% mai mult decât timpul estimat
- aproximativ $\frac{3}{4}$ din proiectele de mare dimensiune nu realizează total funcționalitatea dorită
- aproximativ $\frac{1}{4}$ din proiectele de mare dimensiune sunt anulate
- implică mai mulți oameni și costuri foarte ridicate.

Ingineria programării pune accent pe problemele unor astfel de sisteme.

Ce este un proces software?

- Totalitatea activităților (și a produselor acestora) necesare pentru producerea unui sistem software.
- Presupune 4 categorii de activități structurate:
  - Specificare software: se definește funcționalitatea sistemului software și constrângerile cu care opereaza.
  - Dezvoltare software (Design și Implementare): se produce un sistem software care satisface specificatiile.
  - Validare software: se validează sistemul software pentru a se asigura că aceste corespunde cerințelor clientului.
  - Evolutie (Mentenanță) software: sistemul evoluează pentru a satisface noi cerințe.

Metodele ingineriei software

- O metodă a ingineriei software este o modalitate structurată de dezvoltare a unui sistem software care are ca scop producerea unui sistem de calitate.
- Toate metodele sunt bazate pe ideea de a dezvolta modele ale sistemului și de a folosi aceste modele pentru analiza și design.
- Orice metodă trebuie să cuprindă un limbaj de modelare și un proces asociat acesteia (etapele care se parcurg).

Metodele ingineriei software

- metode orientate pe funcționalitate: încearcă să identifice componentele funcționale de bază ale sistemului.
- metode orientate pe obiect: componente sistemului sunt privite ca obiecte ce interacționează.
- toate aceste metode au fost unificate prin UML(Unified Modeling Language) și Unified Process.

Utilitare CASE

- Utilitarele CASE (Computer-Aided Software Engineering) reprezintă o gamă largă de programe care sunt folosite pentru a sprijini activitățile unui proces software, precum: analiza cerințelor, modelarea sistemului, debugging, testare.
- Categorii de utilitare CASE:
  - utilitare pentru analiză și design: editoare, module de analiză care verifică modelul sistemului conform cu regulile metodei aplicate, generatoare de rapoarte care ajută la producerea documentațiilor, generatoare de cod.
  - utilitare pentru implementare și testare: debuggere, sisteme de analiza a programelor, editoare de programe, generatoare de cazuri de testare.
  - utlitare de planificare: utilitare PERT (Program Evaluation and Review Technique), utilitare de estimare (a costului).
  - utilitare pentru managementul configurației: sisteme de management al versiunii, utilitare de control al schimbărilor.

Scopurile ingineriei software

- satisfacerea cerințelor clienților
- minimizarea costurilor
- evitarea erorilor (de design și analiză, de implementare, etc.)
- producerea de programe ușor de adaptat la noi specificații
- scrierea unei documentații utile celor care folosesc programul dezvoltat

Problemele actuale ale ingineriei software

- specificații inconsistente, incoerente și incomplete
- realizarea unor sisteme care nu pot fi reutilizate
- performanțe scăzute ale produsului final
- inexistența unei metrici pentru testarea calității sistemului
- nici un mijloc pentru a coordona complexitatea
- documentație care nu oferă nici un ajutor efectiv

Concluzii

- Se impunea apariția unei discipline care să standardizeze dezvoltarea sistemelor software, iar aceasta s-a concretizat în ceea ce astăzi numim ingineria programării.
- Mediile de dezvoltare softwre nu pot înlocui gândirea, nu pot garanta o coordonare eficientă a proiectului, nu pot evita erorile umane, nu pot garanta că știm ceea ce dorim să facem.
- Chiar dacă o serie de probleme ale "crizei software" au rămas încă nerezolvate, progresele înregistrate în ultimii ani în domeniul ingineriei software dau speranța că procentul reprezentat de software-ul de calitate va continua trendul ascendent.