Concepte fundamentale ale limbajelor de programare

Factori de calitate ai limbajelor de programare

Curs 01

conf. dr. ing. Ciprian-Bogdan Chirila

Universitatea Politehnica Timisoara
Departmentul de Calculatoare si Tehnologia Informatiei

February 20, 2023

Factori de calitatea ai limbajelor de programare

Curs și laborator

- Curs 66%
  - Ciprian-Bogdan Chirila
  - conferențiar
  - doctorat
    - Universitatea Politehnica Timisoara, Romania
    - Universitatea Nice Sophia Antipolis, Franta
    - Universitatea Jyvaskyla, Finlanda
    - Universitatea Bonn, Germania
  - Moștenire inversă în limbajul de programare Eiffel
  - chirila@cs.upt.ro
  - http://www.cs.upt.ro/ chirila
- Laborator 34%
  - Oana Caus
  - asistent universitar
  - oana.caus@cs.upt.ro

Intalnire cu Bjarne Stroustroup (C++) in Nisa, Franta 2003

Cuprins (1)

- Limbaje de programare
- Definirea și implementarea limbajelor de programare
- Atributele entităților de program
- Transmiterea parametrilor
- Subprograme generice

Cuprins (2)

- Tipuri de date
- Tipuri de date abstracte
- Limbaj de programe orientate pe obiecte
- Structuri de control
- Expresii lambda

Factori de calitatea ai limbajelor de programare

1. Organizare
2. Limbaje de programare
3. Criterii în evaluarea limbajelor de programare
   - Consistenta cu notația uzuală
   - Lizibilitatea
   - Tratarea excepțiilor
   - Verificarea formală automată și detecția erorilor
   - Ortogonalitatea
   - Uniformitatea
   - Scalabilitatea
   - Portabilitatea
4. Eficiența

Limbajul de programare

- Limbajul de programe (LP) este
  - o notație formală ce specifică diferite operații ce se execută (de un calculator)
- În ziua de azi există mai multe limbaje de programare
- Puține sunt utilizate pe scară largă pentru a scrie programe

Locul limbajului de programare în procesul de dezvoltare software

Un produs software complex este dezvoltat de obicei în 5 pași sau faze:

- Analiza cerințelor și a specificațiilor
- Proiectarea software
- Implementarea
- Validarea
- Mentenanța

Faza I: Analiza cerințelor și a specificațiilor

- În timpul analizei, necesitățile utilizatorului sunt concentrate într-o serie de cerințe
- Rezultatul acestei faze este un document ce descrie CE trebuie să facă sistemul
- Nu spune nimic despre CUM se vă face acest lucru
- Evaluarea finală a produsului software se referă la setul de cerințe elaborate în această fază

Faza a II-a: Proiectarea software și specificațiile

- Folosindu-se de cerințe, sistemul software este proiectat în mod corespunzător
- În această fază se fac:
  - Specificațiile proiectului
  - Definirea modulelor
  - Definirea interfețelor

Faza a III-a: Implementarea

- Se realizează în concordanta cu specificațiile
- LP este ales astfel încât să fie cât mai potrivit pentru contextul sistemului dezvoltat
- Câteva criterii sunt luate în considerare:
  - Cât de mult programatorul cunoaște LP
  - Cât de mult trăsăturile limbajului sunt potrivite cerințelor
  - Ce facilitați sunt oferite de IDE (Mediul Integrat de Dezvoltare) pentru codare și testare
  - Ce fel de performante la execuție pot fi atinse prin compilarea sistemului cu limbajul selectat

Faza a IV-a: Validarea

- Se face în fiecare fază a procesului de dezvoltare software
- Se referă la verificarea dacă sistemul respectă specificațiile
- Presupune un proces de testare intens
  - Utilizând seturi multiple de date
  - Rulând pe toate ramurile programului
  - Creând condiții extreme

Faza a V-a: Mentenanța

- După darea în explorare pot apărea erori
  - E nevoie de corecții
- Cazuri posibile:
  - Erori nedescoperite în faza de validare
  - Extinderea programului cu funcționalități noi
  - Optimizarea unor părti de program pentru a obține performante mai bune
  - Schimbări pe platforma harware sau software

Locul limbajului de programare

- Care este impactul limbajului de programare?
- Intervine direct în faza a-III-a, adică în faza de implementare
- Interacționează cu toate celelalte instrumente de dezvoltare
- Este implicat în toate fazele de dezvoltare

Locul limbajului de programare

- Proprietățile limbajului de programare pot afecta:
  - validarea
  - mentenanța
  - proiectarea
- De exemplu:
  - Ascunderea informatiei ca metodă de proiectare și facilitate de limbaj în descrierea datelor abstracte
  - Ascunderea informațiilor implică:
    - Descompunerea sistemelor în module
    - Modulele trebuie să aibă interfețe (seturi de funcții, metode)
    - Accesarea modulelor se face prin interfețe
    - Structura modulelor interne nu este vizibila din exterior
  - Limbajele de programare ce oferă aceste facilități se numesc limbaje de programe orientate-obiect

Factori de calitatea ai limbajelor de programare

1. Organizare
2. Limbaje de programare
3. Criterii în evaluarea limbajelor de programare
   - Consistenta cu notația uzuală
   - Lizibilitatea
   - Tratarea excepțiilor
   - Verificarea formală automată și detecția erorilor
   - Ortogonalitatea
   - Uniformitatea
   - Scalabilitatea
   - Portabilitatea
4. Eficiența

Criterii în evaluarea limbajelor de programare

- LP nu este un scop în sine
- LP trebuie să permită crearea eficienta de software de calitate
- Pentru a defini un LP bun, trebuie să definim ce este un sistem software bun
- Cei mai importanți factori de calitate sunt:
  - fiabilitatea
  - mentenabilitatea
  - eficiența

Cei trei factori de calitate

- Fiabilitatea
  - Funcționarea corectă a sistemului software chiar și în prezenta incidentelor hardware și software
- Mentenabilitatea
  - Capacitatea de a include funcționalități noi sau optimizarea celor existente
- Eficiența
  - Implică oferirea de service optime în contextul unor resurse existențe

Alti factori

- Metode de proiectare
- Instrumente IDE (Medii de Dezvoltare Integrate)
- Algoritmi
- Factori umani
- În cele din urmă, dar nu ultimul este ...**limbajul de programare**

Consistenta cu notațiile uzuale

- Notațiile utilizate în programare trebuie să fie apropiate de notațiile uzuale:
  - Științifice
  - Tehnice
  - Economice
  - etc.
- Programatorul se poate concentra pe semantica programului pentru rezolvarea problemelor și nu pe aspecte de notație
- Mai puține erori
- Productivitate crescută

Lizibilitatea

- Programul trebuie să poată fi citit ușor
- Logica sa trebuie să poată fi dedusă usor din context
- Aspect important pentru programatorii ce trebuie să modifice munca altor programatori
- Pentru lizibilitate crescută, un LP trebuie să aibă
  - Identificatori
  - Cuvinte cheie expresive
  - Facilități de descompunere a programelor

Tratarea excepțiilor

- Importanta pentru crearea de programe sigure
- Secvențe de programe ce pot fi specificate pentru a fi activate când se întâmplă fenomene excepționale
  - Depășiri aritmetice
  - Subdepăsiri aritmetice
  - etc.
- Astfel comportamentul programului devine predictibil

Verificarea formală automată și detecția erorilor

- Definiția limbajului de programare trebuie să permită detecția erorilor la compilare atât cât este posibil
- Redundanța utilă este impusă de majoritatea limbajelor de programare
- Aceeași informație (implicită sau explicită)
  - Este specificată în locuri multiple în program
  - Este verificată la compilare

Verificarea la compilare

- O entitate trebuie să fie în primul rand declarata, apoi utilizată
- Exista corespondente de tipuri între
  - Operanzi și operatori
  - Partea stangă și partea dreaptă a unei atribuiri, etc.
- Corespondența de tip dintre parametrii actuali și cei formali
- Respectarea regulilor de vizibilitate
  - Reguli de domeniu
  - Reguli de import și export a entităților între module
  - Tipuri abstracte
  - Obiecte

Verificarea la compilare

- Nu se pot detecta
  - greșeli de logică de program
  - greșeli semantice
- Nu se poate garanta că
  - un program complet compilabil vă funcționa în concordantă cu specificațiile impuse

Verificarea formală a programelor

- Există metode de a verifica logica programelor în manieră automată
- Necesită o descriere formală a specificațiilor
- Semantica limbajului de programare trebuie să fie compatibilă cu metoda de verificare formală
- Se construiește semantica programului verificat bazată pe semantica limbajului
- Sunt necesare instrumente pentru verificarea și potrivirea dintre specificație și semantica programului

Ortogonalitatea

- limbajul trebuie să fie definit pe baza unor facilitați de bază
- aceste facilități trebuie să poată fi combinate în mod liber
- cu efecte predictibile
- fară restricții
- de exemplu: lipsa de orgonalitate în Pascal
  - funcțiile nu pot fi membri într-un tip structurat

Uniformitatea

- Construcții similare trebuie să aibă semantica similara
- De exemplu: lipsa de uniformitate în limbajul C pentru cuvântul cheie static
  - Utilizat în funcții se refera la alocarea memoriei
    - opusul lui automatic
  - utilizarea în afara funcțiilor influențează vizibilitatea
    - variabila declarată static este vizibilă doar în modulul în care este declarată

Scalabilitatea - facilitatea de a crea programe mari

- Modularizarea programelor
- Ierarhizarea componentelor
- Facilități principale
  - tipuri abstracte
  - module
  - compilarea separată
  - obiecte

Portabilitatea

- Mutarea unui program de pe un calculator pe altul
  - Fară modificări
  - Cu modificări cât mai puține
- idealul "independentei de mașină" este imposibil de atins
- câteva limbaje de programe permit o apropiere de acest deziderat
- Probleme întâmpinate
  - Cuvântul de calculator are lungimi diferite
  - Numerele în virgulă flotantă au reprezentări diferite
  - Operațiile de intrare-ieșire sunt realizate în moduri diferite

Eficiența

- Din punct de vedere al
  - compilării
    - LP trebuie să fie definit astfel încât să faciliteze crearea de compilatoare rapide
  - programului obiect
    - Declararea variabilelor și tipurilor acestora
    - Inferarea tipurilor expresiilor la compilare
    - Tipizare puternica că în limbajul Pascal

Bibliografie

- Brian Kernighan, Dennis Ritchie, C Programming Language, second edition, Prentice Hall, 1978.
- Carlo Ghezzi, Mehdi Jarayeri – Programming Languages, John Wiley, 1987.
- Horia Ciocârlie – Universul limbajelor de programare, ediția 2-a, editura Orizonturi Universitare, Timișoara, 2013.