INGINERIE SOFTWARE

REUTILIZAREA PRODUSELOR SOFTWARE

**Student:Sima Viorel**

**Grupa: 443A**

Cuprins

1. Refolosirea produselor software

2.Apariția de noi metode de refolosire a produselor software

3.Tipuri de refolosire a codului

3.1.Oportunistic

3.2.Confrm unui plan

3.3.Uz intern

3.4.Uz extern

3.5.Referinta

3.6.Bifurcar

4.Exemple

4.1.Librarii software

4.2.Parametrii de design

4.3. Frameworks

4.4. Folosirea sistematica a codului folosit

4.4.Refolosirea softului pentu a putea rula software mai vechi

5.Exemplu de reforlosire a codului

6.Concluzii

7.Bibliografie

**1.Refolosirea produselor software**

De la începutul apariției programării , programatorii au folosit întotdeauna porțiuni de cod ,funcții și proceduri.

Refolosirea softului este recunoscuta a o arie care apartine ingineriei software înca din anul 1968 cand Douglas Mcllroy a propus bazele ingineriei software și al refolosirii componentelor.

Refolosirea codului este gandită pentru a salva timp și resurse lundu-se avantajele codului deja creeat și încercandu-se să se creeze noi programe .Ideea principala este să se foloseasca parți ca sa se construiască un program.

Refolosirea codului implica catrebuie sa se creeze versiuni separate pentur o mai buna mentenanta a codului.În timp ce codul este cea mai refolosita resursa asta nu inseaman ca nu se pot folosi și designs sau documentații teste .

Librăria software este un bun exemplu de cum aceast tip de refolosire a codului afectează noul program.Programatorul poate sa decidă sa creeze librării personalizate. Unele caracteeristici care fac ca codul sa fie mai usor de refolosit este modularitatea , separarea in functii, ascunderea informatiei , separarea pe capitole.

Pentru un cod care doreste sa foloseasca o parte de cod mai vechi are nevoie de o legarura dintre ele și acesta această legătura trebuie sa fie definită.

**2.Apariția de noi metode de refolosire a produselor software**

De obicei se foloseste un program mai vechi si pe baza sa se construieste noul program .Aceasta refolosire de cod de obicei înseamnă o simplă copiere a codului de la un program existent la un nou program .Problemele ce pot sa apara în urma acestui procedeu sunt duplicari de cod care pot duce la avertismente sau la oprirea funcționării programului

Cercetătorii caută sa găsească cea mai rapidă și mai ușoară și sistematica procedura care poate sa ofere foarte rapida adaptare a codului la condițile actuale . Bazele programării obiect orientate au fost pruse din această pricina ca sa se poata refolosi mult mai ușor codul

O alta metodă este aceea de a folosi generatoare de cod , aceste sunt programe care pot creea noi prgrame de un anumit tip bazandu-se pe un set de parametri .

**3.Tipuri de refolosire a codului**

**3.1.Oportunistic**

* atunci cand o echipa începe un proiect observa ca deja o parte din acel proiect exista deja si astfel daca va folosi acel cod munca depusă va fi diminuată

**3.2.Conform unui plan**

* o echipa se gandese te sa facă un program astfel incat acesta sa fie portabil pentu urmatoarele proiecte in care vor fi implicați

**3.3.Uz intern**

* o echipa refoloseste acelasi cod deoarece decizile care au sost luate se aplica si pentru noul proiect

**3.4.Uz extern**

* daca aleg sa ia un program licențiat acet lucru îi va costa , ei trebuie sa se gandeasca daca acest lucru este profitabil sau daca trebuie sa dezvolte aceea alicatie ei înșisi prenru scaderea costurilor

**3.5.Referina**

* codul clientului conține referințe la un cod mai vechi astfel, cand se face proiectarea noului cod trebuie să se țina seama de acest lucru

**3.6.Bifurcarea**

* în interiorul codului clientului se găseste o copie care nu este de tip open source astfel aceasta nu poate fi folosită pentru un nou cilcu de viată al produsului

**4.Exemple**

**4.1.Librarii software**

O foarte folosită metodă de refolosire a codului este tehnica de folosire a librăriilor .Multe operații comune cum ar fi convertirea informatiei folosind diferite formate , manipularea informatiei in moduri comune sunt necesare pentu diferite programe .Autorii noilor programe pot sa foloseasca codul in o librărie software pentru a executa aceste taskuri .

Beneficiile unei astfel de implemententări constă în faptul că această implementare este testata si este dese ori folosită .

Dezavantajele includ inabilitatea de a detalia informatile , un tipm crescut necesar pentu a invata, configura librăriile.

**4.2.Parametrii de design**

Acești parametri sunt de obicei solutia la problema curenta .Acești parametrii sunt mai mult conceptuali decât tangibili si pot fi modificați dupa nevoile programatorului

**4.3.Frameworks**

"În dezvoltarea de software un *framework* este o structură conceptuală și reprezintă o arhitectură de software care modelează relațiile generale ale entităților domeniului (site-ului)".[7]

Dezvoltatorii folosesc de obicei parti mari de programe third-party

**4.4.Folosirea sistematica a codului folosit**

Este o strategie pentru a crește productivitatea și calitatea în industria software .Cu totul ca este un concept simplu , este greu de folosit acest lucru în practică .

Un motiv pentru care acest lucru este greu de implementat este faptul ca există o dependență între software și contextul în care a fost dezvoltat

**4.5.Refolosirea codului pentru a putea rula software mai vechi**

Calculatoarele actuale trebuiesa fie capabile sa ruleze si programe mai vechi acest lucru se poate face cu ajutorul emulatoreleor aceste emulatore au fost create cu ajutorul programelor intiale sub care rula acel program exemplu MS-DOS

**5.Exemplu de reforlosire a codului**

Se doreste sa se adune antitatea totala de produse care va fi livrata unui magazin, programatorul va folosi aceata parte din programa pentru a vedea cum functioneaza limbajul C ++ apoi va generaliza acest cod , si va redenumii variabilele astfel incat sa fie usor de retinut pentru alpicatia sa .

De exeplu a va demeni cantitatea de rosii, b va deveni cantitatea de orez si asa mai departe acest cod il va ajuta pe programator sa poate scrie codul mai rapid facand putine modificari si astfel crescand tipul alaocat pentru alte programe ce trebuiesc scrise ulterior.

"#include<iostream>

#include <conio.h>

using namespace std;

main()

{

int a,b,c;

cout<<"Enter two numbers"<<endl;

cin>>a>>b;

c=a+b;

cout<<"Sum of two numbers are"<<c;

cout<<c<<endl;

getch();

}" nota [6] bibliografie

**6.Concluzii**

In acest studiu de caz despre folosirea codului am putut sa observ cat de rapid se poate scrie un nou program daca , un program asemanator a fost deja lansat .

S-a facut si o mentionare despre DOXBOX acesta este un program care emuleaza ms-dosul problema cu ele consta in faptul ca nu pote fi folosit pentru aplicatii colpicate din aceasta cauza au aprut virtual boxurile care si ele au problema ca nu pot sa recunoasca placa video .

In concluzie codul este refolosit ceea ce duce la o rapiditate in exectarea de noi programe problema este ca este foarte greu sa se faca acest lucru sistematic deoarece aplicatile sunt adaptate pentru nevoile utilizatorului si astfel in unele cazuri este de preferat sa se inceapa allicatia de la 0.

**7.Bibliografie**

[1]http://sharednow.blogspot.ro/2011/05/its-not-just-reuse.html

[2]http://docforge.com/wiki/Code\_reuse

[3]http://en.wikipedia.org/wiki/

[4]http://lombardhill.com/articles/software-reuse-101-what-is-software-reuse/

McConnell, Steve; Rapid Development: Taming Wild Software Schedules, (1996),

[5]http://dl.acm.org/citation.cfm?id=1456662&dl=GUIDE&coll=GUIDE

[6] http://www.cplusplus.com/forum/beginner/96209/

[7] http://ro.wikipedia.org/wiki/Framework