UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Proiect nr. 154/323 cod SMIS – 4428 cofinanțat de prin Fondul European de Dezvoltare Regională “Investiții pentru viitorul dumneavoastră”.

Programul Operațional Sectorial Creșterea Competitivității Economice - POS CCE

UNIUNEA EUROPEANĂ

GUVERNUL ROMÂNIEI

Instrumente Structurale
2007-2013

Platformă de e-learning și curriculă e-content pentru învățământul superior tehnic

Sisteme CAD/CASE

11. Sisteme CASE orientate-obiect

Introducere

- Un sistem CASE modern oferă una sau mai multe metode de analiză şi proiectare, automatizând într-o măsură cât mai mare activitățile desfăşurate în cadrul acestor metode şi asistând parțial sau în totalitate activitățile ciclului de viață al aplicației software, inclusiv managementul ți asigurarea calității.
- În ultimii ani, o serie de sisteme CASE au fost construite sau extinse pe baza diferitelor metodologii orientate-obiect, care s-au impus în ingineria software. Orientarea obiect nu este numai o tehnică de programare ci este, în primul rând, o noua modalitate de a gândi dezvoltarea de software.
- Modelarea orientata obiect reprezinta un nou mod de gândire si de abordare a problemelor utilizând modele si concepte din lumea reala. Astfel, de la modelul de ciclu de viață ‘în cascadă’, s-a ajuns la modelul ‘în spirală’ şi la ‘ingineria software bazată pe modele’ care au avantajul că reduc timpul de proiectare a versiunilor, iar posibilitățile de dezvoltare de noi versiuni şi de testare de către analişti şi utilizatori sunt mai mari. Pe de altă parte, fără o arhitectură orientată obiect, software-ul nu este capabil să exploateze la maxim facilitățile şi avantajele oferite de ultimele dezvoltări ale tehnologiei.

Concepte de proiectare orientată-obiect

- Abordarea orientată-obiect se bazeaza pe accesul la baza de date prin intermediul unor functii special scrise de programator, denumite de obicei **metode**. Un **obiect** reprezinta o construcție de program care combină datele cu un set de metode pentru accesarea şi gestionarea acestora.
- Un obiect poate oferi metode standardizate pentru efectuarea operațiunilor speciale pe datele sale, fara a fi cunoscute specificul si modul in care aceste sarcini sunt îndeplinite. În acest fel, modificările pot fi făcute pe structura internă sau pe metodele aferente unui obiect fără a implica modificarea intregului program. Această abordare poate fi, de asemenea, utilizata pentru a oferi metode standardizate pe diferite tipuri de obiecte.
- Un **program orientat-obiect** conține de obicei diferite tipuri de obiecte, fiecare tip corespunde unui anumit tip de date complexe sau poate fi un obiect din lumea reala sau un concept, cum ar fi un cont bancar, o gestiune, etc.

Metode de analiză si proiectare orientate obiect

- Spre deosebire de metodele traditionale, metodele orientate obiect propun modelarea concomitentă a doua tipuri de structuri, de date si de prelucrări, prin iterarea analizei datelor si comportamentului acestora în sistem, în vederea obtinerii unor clase de obiecte care înglobează atât date cât si functionalităti.
- Cele mai cunoscute metode de analiză si proiectare orientate obiect sunt:
- OOD (*Object Oriented Design*) – dezvoltată de G. Booch;
- OMT (*Object Modeling Technique*) – dezvoltată de James Rumbaugh, Michael Blaha, William Premerlani s.a;
- OOA (*Object Oriented Analysis*) – dezvoltată de Peter Coad si Edward Yourdon;
- OOSE(Object Oriented Software Engineering) – dezvoltată de Jacobson Ivar;

Evoluția instrumentelor de proiectare software

- Evoluția în timp a instrumentelor de proiectare software este, aproximativ, următoarea:
- Metode structurate de analiză şi proiectare (1965)
- Tehnici de modelare a datelor (1970)
- Limbaje de generația a patra pentru gestiunea de date (1975)
- Instrumente de proiectare a specificațiilor software (1980)
- Instrumente pentru prototipizarea interfeței utilizator (1985)
- Instrumente pentru generarea automată a codului (1990)
- Sisteme CASE integrate (1995)
- Sisteme CASE orientate-obiect (2000)
- Sisteme CASE pentru aplicații web(2005)

Caracteristicile unui sistem CASE orientat obiect

- Caracteristicile de bază pentru ca un sistem sau o colecție de instrumente să fie considerat un produs de tip CASE sau un mediu de tip CASE sunt:
- Să ofere facilități grafice puternice pentru a descrie şi documenta software-ul;
- Să fie integrat, astfel încât să permită transmiterea uşoară a datelor între componente;
- Să stocheze informațiile referitoare la software într-un depozit central, permițând accesarea acestora de către toți membrii echipei de dezvoltare;
- Să poată fi utilizat ca bază pentru automatizarea procesului de proiectare a software-ului, utilizând una sau mai multe metode de analiză şi proiectare;
- Să fie reutilizabil pentru dezvoltare de noi sisteme, aplicații şi programe;
- Să poată fi utilizat pe orice platformă hardware, începând cu calculatoare personale pâna la mainframe-uri;
- Să permită realizarea uşoară a interfeței cu utilizatorul final şi expandarea interacțiunii cu acesta;
- Să asigure dezvoltarea de aplicații în limbaje de programare evoluate.

Criterii de evaluare a sistemelor CASE bazate pe UML

- Sistemele CASE orientate obiect folosesc limbajul standardizat UML(*Unified Modeling Language*) pentru dezvoltarea de aplicatii. Criteriile de evaluare ale sistemelor bazate pe UML pot fi împărțite în :
  (Revista Informatica Economica, nr.2-2003)

➢ **Criterii dependente de limbajul de modelare:**

- *Suportul oferit de instrument limbajului de modelare* - se materializează în numărul de concepte puse la dispoziția utilizatorului şi în tipurile de diagrame ce pot fi construite .
- *Suportul pentru adnotări formale textuale* - se referă la posibilitatea utilizării a două formalisme diferite, unul grafic şi altul textual, reprezintă una din trăsăturile definitorii ale limbajului unificat de modelare;
- *Păstrarea consistenței informațiilor între diferite diagrame* -modificarea efectuată într-o anumită vedere trebuie să se reflecte adecvat în celelalte vederi.

➢ **Criterii independente de limbajul de modelare:**

- *Suportul oferit pentru navigarea prin model* - instrumentul trebuie să includă un browser, care să permită navigarea simplă a modelului, găsirea rapidă şi eficientă a entităților proiectului;
- *Generarea automată a codului* – pe baza specificațiilor de proiectare şi implementare;

Criterii de evaluare a sistemelor CASE bazate pe UML-continuare

- *Reverse Engineering şi Round Trip Engineering* – sunt importante în cazul produselor care au fost realizate fără ajutorul instrumentelor CASE, fiindcă uneori acestea nu dispun de documentație de proiectare actualizată .
- S*uport pentru modelarea aplicațiilor bazate pe componente* – este important ca instrumentele CASE să suporte principalele standarde : Enterprise Java Beans (EJB), COM/DCOM, CORBA, platforma .NET etc. ;
- S*uport pentru modelarea datelor* – UML poate modela datele prin intermediul diagramelor de clase. Un aspect important îl reprezintă generarea de cod DDL (un set de comenzi SQL necesare pentru descrierea şi crearea structurii bazei de date în conformitate cu proiectul);
- R*eutilizabilitatea* - se referă la suportul acordat de instrument pentru utilizarea în faza de proiectare a unor componente de biblioteci (clase, componente, şabloane de proiectare sau chiar proiecte – fiecare cu particularitățile lor);
- *Suport pentru design patterns* – se referă la suportul pentru utilizarea unui şablon de proiectare ca model;
- *Documentația generată pentru proiect* - se referă la informațiile care pot fi obținute folosind informația existentă în diagrame şi repository;
- *Schimbul cu alte instrumente* - se referă la posibilitatea unui import a proiectelor realizate cu alte instrumente şi la posibilitatea exportului într-un alt format;
- *Integrarea cu alte instrumente de dezvoltare* – pentru a putea fi folosite facilitățile de *forward* şi *reverse engineering* este necesar să se folosească un sistem CASE adecvat.