­­REFERAT INGINERIE SOFTWARE

Ingineria software este un domeniu ce implică proiectarea, crearea și întreținerea de software aplicând tehnologii și practici din informatică (știința calculatoarelor), managementul proiectelor, inginerie, proiectarea interfețelor și a altor domenii. Ingineria software este o disciplină relativ tânără, care se schimbă constant fiind influențată de apariţia noilor limbaje de programare. Toate acestea determină actualizarea în permanență a programului cu cele mai noi tendinţe urmate de ingineria software.

Activitatea pe care o desfășoară un inginer software defineşte în mare măsură planificarea şi gestionarea dezvoltării produsului software, aplicând principiile ingineriei software. Aceste principii definesc cum colectăm cerințele cumpărătorilor produsului software, determină arhitectura şi designul care se va aplica, precizează paşii din faza de dezvoltare şi, în mod obligatoriu, stabilesc criteriile unui produs rezistent și de calitate.

Colectarea şi analizarea cerințelor software este o fază foarte importantă în dezvoltarea software-ului. Aceasta este prima fază pe baza căreia se iau mai departe decizii legate de design, arhitectură, planificare estc. Cerințele definesc nevoile cumpărătorului de software, definesc condiţiile de rezolvare a problemelor, determină regulile de afaceri care trebuie aplicate în software, precizează standardele organizaţiei interne, precum şi normele legislative pe care trebuie să le respecte software-ul.

Cu cât cerințele software au o calitate mai înaltă, cu atât crește și calitatea managementului dezvoltării. Printre altele, cerințele permit clientului să vadă ceea ce cere de la software-ul său și pe această bază își poate planifica şi costurile, deoarece primeşte propuneri privind soluţia şi costurile.

Unified Modeling Language (prescurtat UML) este un limbaj standard pentru descrierea de [modele](https://ro.wikipedia.org/w/index.php?title=Modelare_orientat%C4%83_pe_obiect&action=edit&redlink=1" \o "Modelare orientată pe obiect — pagină inexistentă) și [specificații](https://ro.wikipedia.org/w/index.php?title=Specifica%C8%9Bie&action=edit&redlink=1" \o "Specificație — pagină inexistentă) pentru [software](https://ro.wikipedia.org/wiki/Software). Limbajul a fost creat de către consorțiul [Object Management Group](https://ro.wikipedia.org/w/index.php?title=Object_Management_Group&action=edit&redlink=1) (OMG) care a mai produs printre altele și standardul de schimb de mesaje intre sisteme [CORBA](https://ro.wikipedia.org/w/index.php?title=CORBA&action=edit&redlink=1). UML a fost la bază dezvoltat pentru reprezentarea complexității programelor orientate pe obiect, al căror fundament este structurarea programelor pe clase, și instanțele acestora (numite și obiecte). Cu toate acestea, datorită eficienței și clarității în reprezentarea unor elemente abstracte, UML este utilizat dincolo de domeniul IT. Așa se face că există aplicații ale UML-ului pentru management de proiecte, pentru *business Process Design* etc.

UML oferă o largă gamă de diagrame pentru modelarea diferitelor situații în cadrul unui proiect de dezvoltare software.

Diagramă de activitate  
Așa cum și numele sugerează, acest tip de diagrame UML este folosit pentru reprezentarea desfășurării unei activități în rularea programului.

### Diagramă de componente

Sunt diagrame ce se utilizează în cadrul proiectării unei [arhitecturi de sistem](https://ro.wikipedia.org/w/index.php?title=Arhitectur%C4%83_de_sistem&action=edit&redlink=1" \o "Arhitectură de sistem — pagină inexistentă), atunci când diferite componente ale sistemului software (numite și subsisteme) interacționează între ele prin intermediul unor [interfețe](https://ro.wikipedia.org/w/index.php?title=Interfa%C8%9B%C4%83_(Informatic%C4%83)&action=edit&redlink=1" \o "Interfață (Informatică) — pagină inexistentă).

Diagramă de clasă

O diagramă de clasă reprezentând multiplicitatea dintre două clase Este folosită pentru reprezentarea vizuală a claselor și a interdependențelor, [taxonomiei](https://ro.wikipedia.org/wiki/Taxonomie" \o "Taxonomie) și a relațiilor de [multiplicitate](https://ro.wikipedia.org/w/index.php?title=Multiplicitate&action=edit&redlink=1" \o "Multiplicitate — pagină inexistentă) dintre ele. Diagramele de clasă sunt folosite și pentru reprezentarea concretă a unor instanțe de clasă, așadar obiecte, și a legăturilor concrete dintre acestea.

### Diagramă package

Este folosită pentru reprezentarea interdependențelor și a relațiilor dintre pachete ce formează un program. (Pachetele reprezintă în acest sens containerele ce conțin clase și sunt o caracteristică a unor limbaje de programare ca [Java](https://ro.wikipedia.org/wiki/Java) sau [C Sharp](https://ro.wikipedia.org/wiki/C_Sharp))

Diagramă de secvență

Este utilizată în special în timpul dezvoltării de subsisteme, și ajută la identificarea relațiilor dintre obiecte pe parcursul efectuării unei anumite operații (de exemplu relația dintre obiectul client Ion și obiectul comandă numărul 1234 în timpul operației de finalizare a unei comenzi).

Datorită simplității ei este utilizată în special în discuțiile dintre firma dezvoltatoare de software și clienți sau utilizatori. Diagramele use case reprezintă interacțiunea dintre elementele exterioare unui sistem (numite și actori) și sistem. În cazul acestor diagrame se prezintă acțiunea desfășurată de sistem la interacțiunea actorului, însă modul în care sistemul desfășoare acea acțiune nu trebuie să fie reprezentat într-o astfel de diagramă (conceptul [blackBox](https://ro.wikipedia.org/w/index.php?title=BlackBox_(Informatic%C4%83)&action=edit&redlink=1" \o "BlackBox (Informatică) — pagină inexistentă))

.

### Diagramă deployment

Este folosită în cadrul proiectării arhitecturii de sistem, având scopul reprezentării repartizării fizice a diferitelor elemente ale sistemului (de exemplu repartizarea unui sistem ce este compus dintr-un [server](https://ro.wikipedia.org/wiki/Server), [routere](https://ro.wikipedia.org/wiki/Router" \o "Router) și alte componente exterioare, inclusiv [software](https://ro.wikipedia.org/wiki/Software)).