INTRODUCERE

Termenul de Software Engineering a fost folosit prima data la o conferinta organizata de NATO, in 1968, la Garmish, pentru a se referi la activitatile legate de producerea in maniera industriala a software-ului. Scopul conferintei: incercarea de a iesi din starea dezolanta in care se afla software-ul. Problema software-ului s-a pus in momentul in care programele incepusera sa aiba multe linii de cod.

Ingineria software este un domeniu de studiu tehnologic, aplicarea unei strategii **sistematice**, **disciplinate** si **masurabile** pentru **dezvoltarea**, **utilizarea** si **intretinerea** software-ului.

1.1 Procesul software si Ingineria software

Procesul software (sau proces de dezvoltare a software-ului) este un **proces** care conduce la realizarea unui produs software. In general, un **proces** este o **activitate** complexa realizata de oameni si/sau masini cu scopul de a produce unul sau mai multe produse de iesire, plecand de la niste elemente de intrare: date, materia prima, etc. O **activitate** este o colectie de task-uri realizate cu un anumit scop.

Procesul software functioneaza ca un cadru de referinta in care se desfasoara toate activitatile necesare obtinerii unui produs software de calitate.

Cadru de referinta = un numar mic de activitati comune tuturor proceselor software:
1. faza de definire
- structurarea sistemului sau a informatiilor
- planificarea proiectului
- analiza cerintelor

2. faza de dezvoltare
- proiectarea software-ului
- generarea codului
- validarea software-ului

3. faza de intretinere

Activitatile auxiliare:
- conducerea si controlul procesului
- reviziile tehnice formale
- garantarea calitatii produsului
- gestiunea configuratiilor software
- documentarea
- gestiunea elementelor reutilizabile
- gestiunea riscurilor
- masuratorile de performanta

se suprapun peste modelul procesului.Acestea sunt independente de activitatile de bază si se desfasoara pe tot parcursul ciclului de viata al procesului software.

O colectie de task-uri (sarcini), ca: puncte de control, produse intermediare, puncte de garantie a calitatii, permit adaptarea activitatilor portante la caracteristicile proiectului software respectiv si la exigentele echipei de dezvoltare.

Asadar, procesul software defineste **strategia** adoptata pentru producerea software-ului, dar nu si **tehnologiile** (metode, instrumente si tehnici) utilizate in cadrul procesului. De tehnologii se ocupa ingineria software.

Ingineria software este o **tehnologie stratificata** pe 4 nivele:
1. **Grija pentru calitate**. Sta la baza ingineriei software si duce la imbunatatirea progresiva a procesului software si produce strategii tot mai mature pentru ingineria software.
2. **Procesul software**.
3. **Metode**.

Definitie. O **metoda** este un proces disciplinat, orientat spre generarea unui ansamblu de modele care descriu diferitele aspecte ale unui sistem de dezvoltare, utilizand un limbaj propriu, bine-definit, eventual instrumente care faciliteaza procesul de modelare.

Metodele constituie cunostintele tehnice legate de constructia software-ului. Acestea se refera la o gama larga de activitati care cuprind activitatile portante. De obicei, metodele sunt grupate in cadrul unei **metodologii = colectie de metode folosite pentru rezolvarea unei clase de probleme, aplicabile in timpul dezvoltarii unui sistem si grupate intr-o anumita abordare generala, filozofica.**

1. **Instrumente.** Ofera suport automatizat proiectului si metodelor. Cand sunt **integrate mai multe instrumente**, adica datelele furnizate de un instrument sunt imediat utlizate de un alt instrument se obtine un suport pentru dezvoltarea software-ului numit CASE (Computer Aided Software Engineering), similar sistemelor CAD/CAE.

1.2 Zonele (domeniile) cheie ale procesului software

Pentru un mai bun **control** al procesului software si pentru distributia eficienta a tehnologiei software trebuie instituite si gestionate **zonele cheie** (KPA – Key Process Area). Zonele cheie formeaza baza controlului de gestiune a proiectelor software si stabilesc unde trebuie aplicate anumite metode sau tehnici, care sunt produsele intermediare care trebuie scrise (produse), care sunt punctele de control (**milestone-uri**), cum sa fie garantata **calitatea** si cum sa fie administrate in modul cele mai oportun **modificarile.**

Zonele cheie descriu **functiile** (de exemplu, planificarea procesului, gestiunea cerintelor) care trebuie sa fie indeplinite pentru a garanta un anumit **nivel de maturitate** a produsului final.

Conceptul de **nivel de maturitate** indica nivelul de profesionalitate si calitate a produselor unei firme specializate in producerea software-ului. Modelul de capacitate matura (Capability Maturity Model - CMM) al Institului de de Ingineria Software (ISE-1986) furnizeaza o strategie de imbunatatire a procesului si este format din urmatoarele 5 nivele:

1. **initial** – procesul software este definit de fiecare data, de multe ori confuz
2. **repetabil** – se folosesc procese de baza de gestiune a proiectelor precedente (costuri, durata, functionalitate, resurse). Acestea sunt necesare pentru a se repeta reusitele precedente*;* poate fi folosit pentru preziceri in proiectul urmator; actiune imediata de corectare a problemelor identificate
3. **definit –** procesul software este complet documentat conform unui standard si inclus in procesul software la nivelul intreprinderii;aspectele manageriale si tehnice clar definite; eforturi continue de a imbunatati calitatea, productivitatea; revizii pentru a imbuntatii calitatea; acum se folosesc instrumente CASE
4. **gestionat –** scopurile fiecarui proiect sunt calitatea si productivitatea. Acestea sunt continuu monitorizate; controale statistice a calitatii;
5. **optimizat -** imbunatatirea continua a procesului; controlul procesului si controale statistice a calitatii; folosirea cunostiintelor folosite in proiectele anterioare

In practica, indicatorul real al maturitatii procesului este nivelul de predictibilitate in performanta (cost, planificare, calitate) proiectului. Coreland performanta proiectului cu cele 5 nivele de maturitate avem urmatoarele caracteristici:

Nivelul 1 – are o performanta aleatoare (nepredictibila)

Nivelul 2 – atinge o performanta ce se poate repeta de la un proiect la altul

Nivelul 3 – arata o performanta mai buna pe proiecte succesive in termeni de cost, planificare sau calitate

Nivelul 4 – demonstreaza faptul ca performanta se imbunatateste pe proiecte ulterioare substantial intr-o dimensiune a performantei sasu semnficativ pe mai multe dimensiuni (eg, pe cost si calitate)

Nivelul 5 – corespunde unei performante foarte inalte pe proiecte ulterioare sau imbuntatiri substantiale in toate dimensiunile.

Scopul majoritatii firmelor specializate este de a atinge nivelul 3. Dar un proces de nivel 3 nu inseamna neaparat un proces bun. Pe de alta parte, un proces intr-adevar bun ar trebui sa atinga cu usurinta nivelul 3. Indicatorii unui proces adevarat sunt:

- intelegerea obiectiva a nivelului curent de maturitate
- intelegerea obiectiva a performantei proiectului in termeni de cost si calitate
- imbunatatirea reala a performantei proiectului
- timpul minim necesar pregatirii pentru o evaluare

1.3 Modele ale procesului software. Procesul software conventional

Ciclul de viata al unui produs software este un model al activitatilor de dezvoltare si operare al sistemului informatic bazat pe software. In sens larg poate semnifica si o metodologie pentru dezvoltarea unui sistem, si apoi pentru operarea cu acesta.

Alegerea modelului de proces software este o decizie strategica pentru o firma care produce software, deoarece modificarile de organizare, staff si activitatile interne necesare implementarii procesului pot influenta pe termen lung productivitatea firmei. Aceasta alegere depinde de urmatoarele elemente:

- natura proiectului si a aplicatiei de realizat,
- competentele specifice si experienta acumulata in proiectele precedente de membrii echipei de dezvoltare,
- metodele si instrumentele care vor fi utilizate,
- controalele si produsele cerute.

Exista numeroase modele sau cicluri de viata. Ele pot fi clasificate in cateva tipuri. Cele mai cunoscute tipuri de modele sunt: tipul clasic, “in cascada”, tipul structurat si, cel mai recent, tipul incremental. In realitate, fiecare metodologie de dezvoltare a unui sistem propune un ciclu propriu de viata, care este apoi adaptat la exigentele firmei care il utilizeaza.

Organizarea unui proiect dupa un anumit model nu este altceva decat un mod unic de a actiona, comun tuturor persoanelor care participa la dezvoltarea produsului sau sistemului. Modelul poate organiza activitatile managerilor proiectului, facand mai probabil ca problemele juste vor fi abordate la momentele juste. Dorinta de a avea un ciclu de viata al proiectului rezulta din trei obiective principale ale oricarui producator de software:

- definirea activitatilor ce trebuie desfasurate intr-un proiect de dezvoltare a sistemelor;
- introducerea coerentei intre numeroasele proiecte de dezvoltare a sistemelor cu aceeasi organizare;
- furnizarea de puncte de control (checkpoint) destinate management-ului in vederea luarii de decizii de tipul *go/no-go*.

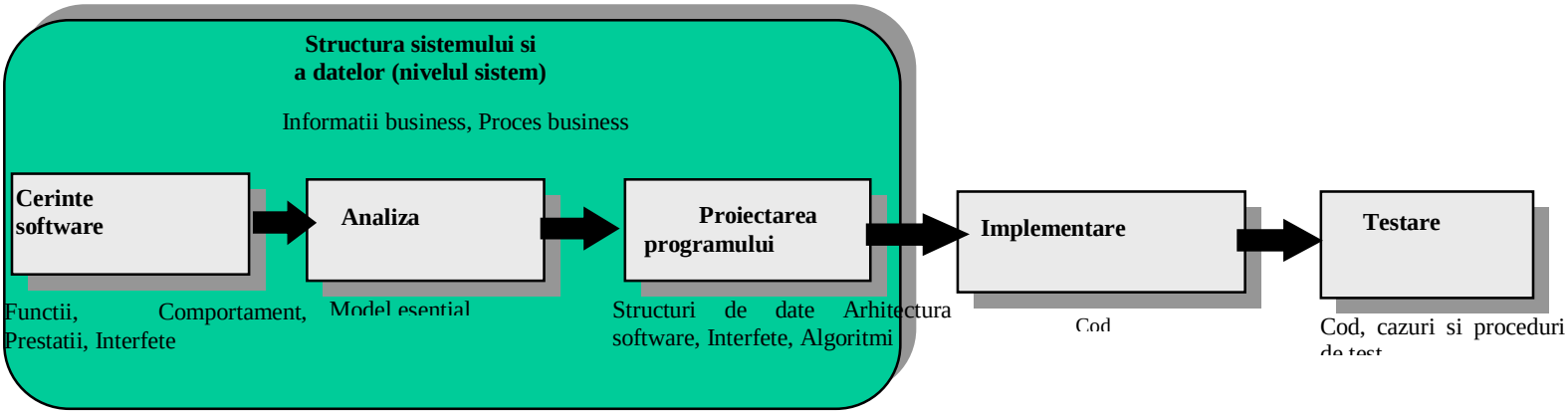
Pe de alta parte, ciclul de viata este consecinta alegerii unei strategii globale ce trebuie urmata pentru a construi un sistem informatic. Asa cum am prezentat in Figura 1-1, pentru a construi un sistem informatic pentru o problema din lumea reala trebuie sa ne deplasam din universul problemei in spatiul solutiilor bazate pe calculator. Cum sa se proiecteze lumea reala in spatiul solutiilor bazate pe calculator este subiectul diferitelor strategii existente pentru dezvoltarea sistemelor informatice.

1.3.1 Modelul secvential liniar: modelul in cascada

Acesta are mai multe **faze**, care sunt executate pe rand (in cascada). Rezultatul unei faze reprezinta intrarea fazei urmatoare. Dificultatea principala din acest model este lipsa paralelismului intre activitati. Modelul are o tendinta acuta spre implementarea bottom-up a sistemului (programatorii executa mai intai testul tuturor modulelor lor, apoi testul subsistemelor si in fine testul sistemului, procedeu cunoscut ca “ciclu de viata in cascada” [1],[2].) si o insistenta (accentuare) pe evolutia liniara secventiala de la o faza la urmatoarea.

Fazele modelului in cascada:

[1] Winston W. Royce, “Managing the Development of Large Software Systems”, Proceedings, IEEE Wescon, august 1970, p. 1-9.
[2] Barry Boehm, “Software Engineering Economics”, Englewood Cliffs, N.J.: Prentice Hall, 1981.



1. Capturarea cerintelor software. Scopurile acestei faze:

- trebuie sa determine ce doreste clientul
- trebuie sa determine nevoile clientului

Tehnicile folosite in aceasta faza:

- interviuri (tehnica principala)
- interviuri structurate si nestructurate
- chestionare
- formulare de analiza
- camere video
- scenarii

Actori: utilizatori, consultanti externi, eventual analistul

Se obtine documentul proiectului sau studiul de fezabilitate plus planul proiectului.

Ciclul de viata clasic este condus de cerinte, ceea ce implica definirea precisa, completa si neambigua a cerintelor si apoi implementarea exacta a acestora. Si asta inainte ca celelalte faze sa inceapa. De aceea specificarea cerintelor este o parte importanta si dificila a procesului software.

Apoi, cerintele sunt de obicei specificate in **maniera functionala**, insusi software-ul este descompus in functii; cerintele sunt alocate apoi componentelor rezultate. Deseori aceasta descompunere este foarte diferita de o descompunerea bazata pe proiectarea OO si folosirea componentelor existente. Descompunerea functionala (algoritmica, bazate pe DFD) devine astfel ancorata in contracte, subcontracte si structuri de lucru.

2. Analiza

Scopul analizei este transformarea cerintelor utilizatorului si documentul proiectului intr-o specificatie structurata a sistemului si intr-o proiectare a bugetului necesar dezvoltarii. Specificatia reprezinta un model conceptual al sistemului si este alcatuita din mai multe sectiuni corespunzatoare diverselor puncte de vedere asupra sistemului: modelul static, modelul comportamental, modelul functional etc.

3. **Proiectarea**

In cadrul proiectarii se ating patru obiective importante pentru dezvoltarea unui produs informatic: structura datelor majore ale sistemului, arhitectura software, interfetele dintre componentele software ale sistemului si algoritmii principali folositi in cadrul componentelor. Proiectarea se dezvolta in doua subfaze.

Activitatea de proiectare intr-o prima subfaza a sa se ocupa de alocarea unor portiuni ale specificatiei sistemului atât procesoarelor din sistem, cât taskurilor existente in cadrul fiecarui procesor. Aceasta subfaza a proiectarii se numeste *proiectare arhitecturala.*

In interiorul fiecarui task, activitatea de proiectare realizeaza o ierarhie de module de program si de interfete intre module necesara pentru implementarea specificatiilor rezultate din analiza. Aceasta subfaza este *proiectarea detaliata*.

In paralel, proiectarea se ocupa de transformarea modelului sonceptual al sistemului, in special al modelului static, intr-un proiect al bazei de date a sistemului.

4. **Generarea de cod si implementarea**

Aceasta faza cuprinde atat scrierea codului pentru modulele sistemului, cat si integrarea modulelor intr-un schelet din ce in ce mai complet al sistemului final.

5. **Testarea de unitati de program**

Este o activitate care priveste verificarea modulelor de program.

6. **Verificarea subsistemelor**

Subsistemele sunt create din modulele componente, apoi sunt testate si verificate din punct de vedere functional.

7. **Validarea sistemului**

Este activitatea finala care cuprinde, evident, integrarea sistemului, acceptarea din partea clientului si instalarea sistemului in mediul sau. Validarea se concentreaza asupra verificarii aspectelor logice interne ale sistemului in vederea garantarii ca toate functiile sistemului au fost implementate in conformitate cu cerintele.

In acest punct, sistemul trebuie sa functioneze conform cerintelor utilizatorului, actualizate sau nu in timpul dezvoltarii.

8. **Intretinerea**

Este activitatea cea mai de lunga durata si mai costisitoare din evolutia oricarui sistem. In aceasta faza sistemul se prezinta ca o entitate dinamica in continua modificare. Modificarile se datoreaza fie ajustarii lui pentru a corespunde cerintelor initiale prin eliminarea bug-urilor, cat si adaptarii lui la noi cerinte ce provin din partea clientului.

Probleme:

- proiectele reale nu respecta schema secventiala
- orice modificare in faza de proiectare determina modificarea cerintelor
- cerintele incomplete nu pot fi guvernate
- greselile logice cele mai banale sunt descoperite la inceputul perioadei de testare, in timp ce cele mai serioase sunt gasite ultimele
- versiune care sa functioneze se poate obtine doar la sfarsitul proiectului
- apar “stari de blocaj” datorate necesitatii de sincronizare intre activitati sau intre membrii echipei (adversitati intre skateholder-i datorate dificultatilor de specificare a cerintelor si a schimbului de informatii numai prin intermediul documentelor pe hartie, in formate ad-hoc. Lipsa unei notatii riguroase genereaza revizii subiective)

Totusi, modelul “in cascada” ocupa un loc bine definit si important in ingineria programarii, fiind modelul cel mai raspandit. Este considerat ca fiind un standard (benchmark) al procesului software conventional.

Modificari ale modelului care pot imbunatati rezultatele procesului:

- ***Proiectarea arhitecturala ar trebui sa preceada analiza si proiectarea de detaliu***. Prin aceasta tehnica, proiectantul se asigura ca software-ul nu va esua datorita stocarii, temporizarii si a fluxului de date. Cum faza urmatoare este analiza, priectantul trebuie sa impuna analistului constrangerile operationale, de stocare si temporizare. Proiectarea arhitecturala se incheie cu un document informativ si la zi astefl incat oricare membru al echipei va intelege in mod elementar cum functioneaza sistemul.
- ***Documentarea continua si completa a activitatilor si produselor***
- ***Repetarea unor activitati daca se dovedeste necesar***. Generalizarea ar fi: repeta de ori de cate ori este nevoie => principiu al dezvoltarii iterative moderne.
- ***Planificarea, controlarea si monitorizarea testarii***.Cum?
    a. cu ajutorul unei echipe de testare indipendente

b. gasirea erorilor logice evidente - astazi este depasit, datorita folosirii compilatoarelor, analizoarelor si altor instrumente de captare a erorilor evidente

c. se testeaza fiecare cale logica – astazi se aplica numai sistemelor distribuite unde, cum timpul este o dimensiune in plus, exista un numar infinit de cai logice.

- ***Implicarea clientulului in dezvoltare*** in 3 momente inainte de furnizarea finala:

a. revizie software preliminara, dupa subfaza de proiectare arhitecturala

b. o secventa de revizii in timpul proiectarii detaliate

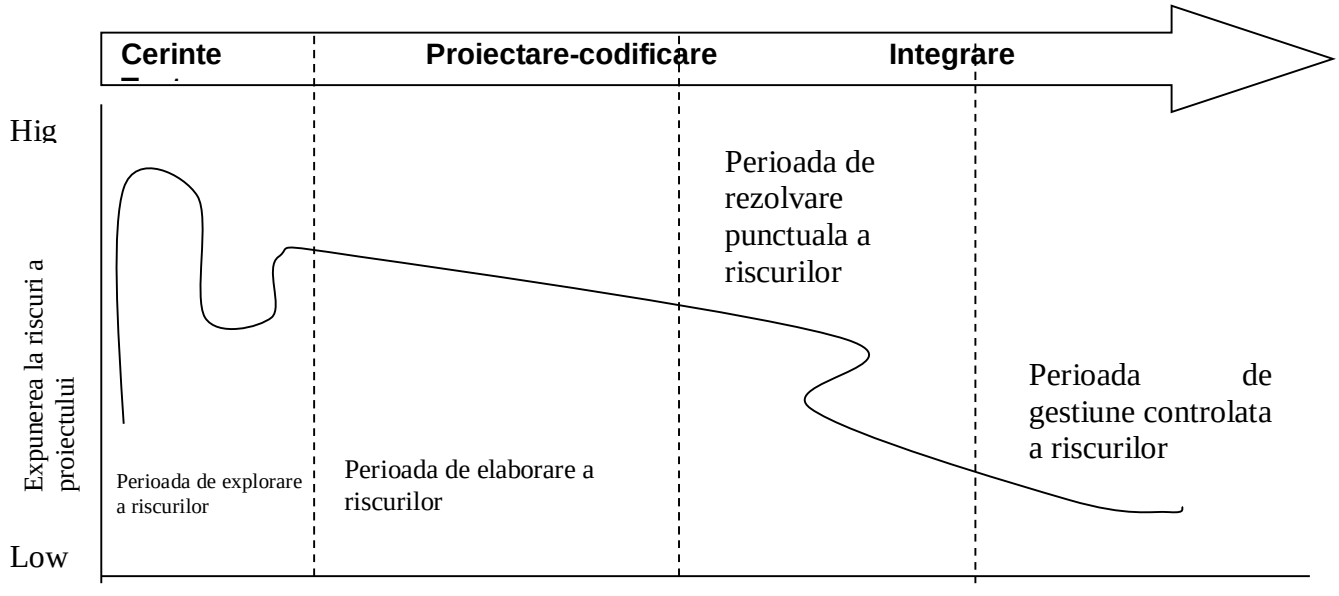
c. revizie finala de acceptare a produsului software, dupa faza de testare

d. este o tehnica valabila si astazi: implicarea clientului prin intermediul demonstratiilor si a predarilor alfa/beta planificate

Cheltuielile/activitate pentru un proiect conventional software

Activitate	**Cost**
Management	5%
Capturarea cerintelor	5%
Proiectare	10%
Codificarea si testarea unitatilor de program	30%
Integrarea si testarea	40%
Repartitia	5%
Mediu	5%

Profilul riscului pentru proiectele ce folosesc acest model



Ciclul de viata al proiectului

Definitie. Riscul este definit ca probabilitatea lipsei unui scop de cost, plan, caracteristica sau calitate.

La inceputul ciclului de viata, dupa ce au fost identificate cerintele, expunerea la riscuri este foarte mare. Dupa ce a fost facut un concept de proiectare, chiar si numai pe hartie pentru a se intelege riscurile, expunerea la riscuri se stabilizeaza. Totusi, aceasta stabilizare se face la un nivel relativ inalt, deoarece managerul software are prea putine fapte tangibile pentru a obtine o evaluare obiectiva. In timpul codificarii, sunt rezolvate unele componente cu risc mare. De abia in faza de integrare, calitatile si riscurile reale ale sistemului devin tangibile. In aceasta perioada sunt rezolvate multe probleme de proiectare si se fac compromisuri ingineresti (trade-off). De aceea proiectele tind sa aiba o faza de integrare extinsa.

1.3.2 Modelul RAD (Rapid Application Development)

Modelul RAD (*Rapid Application Development*) este un model secvential liniar care este orientat spre un ciclu de dezvoltare foarte scurt. Se bazeaza pe strategii constructive bazate pe utilizarea componentelor. Cu cerinte precise si clare, procesul RAD permite mai multor echipe de dezvoltatori sa creeze un sistem intr-un interval de timp foarte scurt. Este utilizat mai ales pentru dezvoltarea unor sisteme informatice de gestiune a intreprinderilor. Conform schemei din figura 1-1., strategia RAD cuprinde urmatoarele faze:

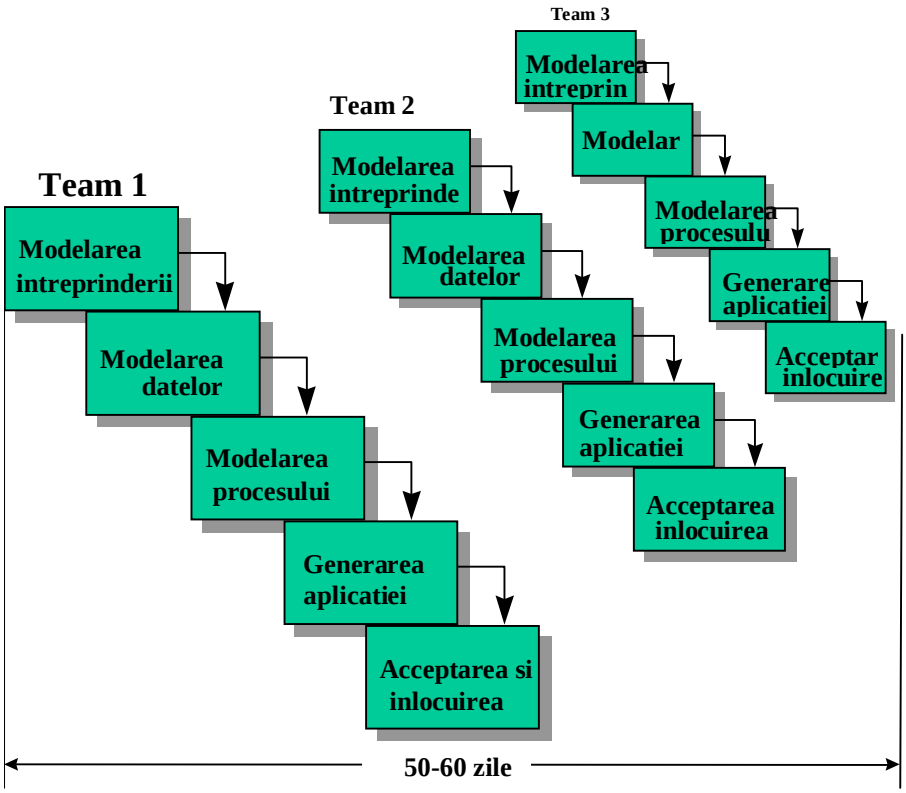


Figura 0-1.Modelul RAD

Modelarea intreprinderii. Sunt identificate datele care conduc *procesul intreprinderii* (*business process*), datele generate de proces si cine le genereaza, datele prelucrate si cine le prelucreaza.

Modelarea datelor. Fluxul de date in intreprindere este rafinat intr-o serie de “obiecte” de suport informational al intreprinderii care sunt descrise.

Modelarea procesului. Obiectele identificate sunt transformate pentru a realiza in final fluxul de informatii necesar pentru implementarea unei functii a business-ului.

Generarea aplicatiei. Se utilizeaza tehnicile de a patra generatie care permit reutilizarea componentelor existente, deja validate, si crearea de noi componente reutilizabile.

Validare. Cum multe componente sunt deja validate, se reduce timpul dedicat validarii. Raman de validat noile componente si interfetele.

RAD se aplica atunci cand functiile primare ale intreprinderii pot fi separate si descrise in foarte scurt timp. Acestea sunt incredintate unor echipe diferite si apoi combinate in produsul final.

Problemele modelului RAD sunt urmatoarele:

- Necesita numeroase resurse umane atunci cand proiectul este de dimensiuni mari.
- Daca angajamentul clientilor sau dezvoltatorilor de a realiza sistemul intr-un timp scurt nu se concretizeaza in fapte, proiectul esueaza.
- Nu functioneaza pentru sisteme care nu pot fi descompuse usor.
- Nu reuseste intotdeauna sa realizeze sisteme cu performante deosebite.
- Nu trebuie utilizat atunci cand noua aplicatie exploateaza tehnologii noi sau prevede un inalt grad de interoperabilitate cu programele existente.

1.3.3 Modelul in spirala

Modelul in spirala a fost propus de Boehm. Este un model de proces software care imbina natura iterativa a prototipizarii si aspectele sistematice si disciplinate ale modelului “in cascada”. El permite o dezvoltare rapida a versiunilor din ce in ce mai complete ale software-ului. Se pleaca de la un model pe hartie sau de la un prototip pentru a se ajunge la versiunea completa a sistemului.

Modelul in spirala se imparte in *zone de activitati* (*task regions*), intre trei si sase. In Figura 0-2 este prezentat un model format din sase zone de activitati.

Comunicarea cu clientul: activitatile care stabilesc o comunicare eficace intre client si dezvoltator;

Planificare: activitatile care definesc resursele, scadentele si alte informatii legate de proiect;

Analiza riscurilor: activitatile care estimeaza riscurile tehnice si de gestiune;

Structurarea: activitatile care construiesc unul sau mai multe modele ale aplicatiei;

Construire si livrare: activitatile care construiesc, valideaza si instaleaza produsul, si care ofera suport utilizatorului;

Evaluarea din partea clientului: activitatile care se ocupa de reactiile clientului, pe baza evaluarii modelului elaborat in faza de structurare si a software-uluigenerat in faza de construire.

Fiecare zona de activitate este alcatuita din sarcini specifice, adaptate caracteristicilor proiectului. In toate cazurile se recurge si la activitatile auxiliare.

De la inceputul procesului evolutiv, echipa de dezvoltare se deplaseaza de-a lungul spiralei in sensul acelor de ceasornic, plecand din centru. Fiecare rotatie da un rezultat: mai intai o specificatie a produsului, apoi un prototip si apoi versiuni din ce in ce mai sofisticate ale software-ului. Fiecare trecere prin zona de planificare aduce ajustari planului proiectului. Costurile si scadentele sunt reevaluate, in functie de reactiile si evaluarile clientului. Seful proiectului stabileste numarul de repetitii prevazute pentru completarea proiectului.

Modelul in spirala poate fi adaptat astfel incat sa se extinda pe intreaga durata de viata a produsului. Se poate pleca de la un model din orice punct al axei punctelor de intrare in proiect (axa care separa evaluarea clientului de comunicatia cu clientul). Spirala ramane operativa pana la terminarea software-ului.

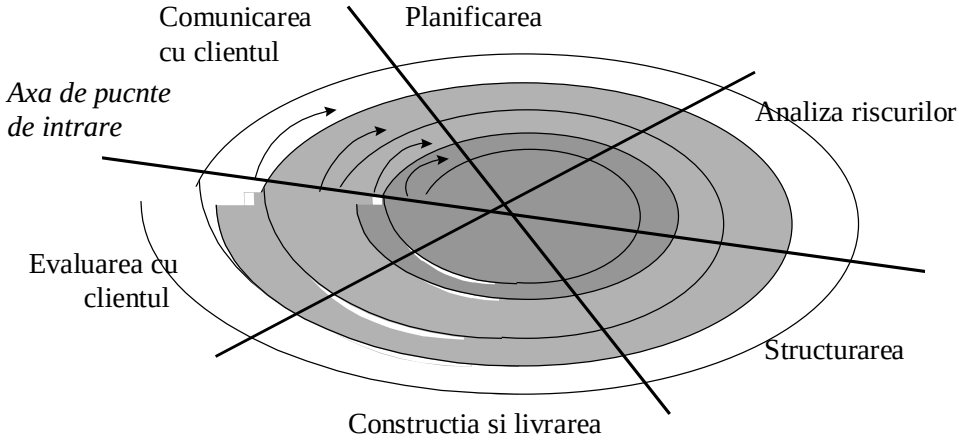


Figura 0-2. Modelul in spirala

Modelul in spirala este o strategie realista pentru dezvoltarea de sisteme si de software de dimensiuni mari. Deoarece software-ul se dezvolta odata cu avansarea procesului, dezvoltatorul si clientul au un mod mai bun de evaluare a riscurilor la fiecare etapa a evolutiei.

Problemele modelului in spirala sunt urmatoarele:

- Este dificil sa se convinga clientii ca o dezvoltare de acest fel poate fi tinuta sub control;
- Necesita competente de nivel inalt pentru estimarea riscurilor;
- Daca un risc important nu este descoperit si controlat la timp, problemele pot creste ulterior exponential;
- Modelul este relativ nou si deci nu este inca evaluat cu precizie.

1.3.4 Modelul de proces software RUP

Caracteristica cea mai discrimanta a unui proces software este separarea bine definita a activitatilor de "cercetare si dezvoltare" de activitatile de "productie". De obicei un proces software esueaza datorita supraccentuarii unuia din cele 2 seturi de activitati.

Modelul procesului software RUP este format din 2 etape:

- **etapa inginereasca**, condusa de echipe mai mici, dar mai putin predictibile, care realizeaza activitatile de analiza, proiectare si planificare
- **etapa de productie**, condusa de echipe mai mari, dar mai mult predictibile, care realizeaza activitatile de implementare, testare si repartitie (deployment).

Diferentele dintre cele 2 etape:

Aspectul ciclului de viata	**Etapa inginereasca**	**Etapa de productie**
Reducerea riscului	Planificare, fezabilitate tehnica	Cost
Produse	Arhitectura baseline	Produse furnizate
Activitati	Analiza, proiectarea, planificarea	Implementare, testare
Evaluare	Demonstratie, inspectie, analiza	Testare
Management	Planificare	Operatii

Aceasta impartire este prea aspra si prea simpla pentru majoritatea aplicatiilor. De accea, etapa inginereasca este descompusa in 2 faze:

- faza de conceptie
- faza de elaborare
- iar etapa de productie la fel in 2 faze:

- faza de constructie
- faza de tranzitie

Asadar, ciclul de viata al unui proces software modern este o trecere completa prin cele 4 faze ⇔ timpul dintre inceputul fazei de conceptie si sfarsitul fazei de tranzitie.

Faza de conceptie

Obiective generale:

- Stabileste granitele sistemului si a conditiilor pe frontiera (cum se opereaza, criteriile de acceptare, intelegere clara a ce trebuie si ce nu trebuie sa fie in produs)
- Identifica impreuna cu clientii si experti in domeniu cazurile de utilizare critice si scenariile fundamentale
- Propune una sau mai multe solutii pentru arhitectura sistemului
- Planifica dezvoltarea (estimeaza costurile si termenele intregului proiect), planifica in detaliu faza de elaborare
- Identifica si grupeaza riscurile (necunoscutele, ceea ce face nepredictibila dezvoltarea)

Activitati principale:

Formularea scopului proiectului. Aceasta activitate implica capturarea cerintelor si a conceptului operational intr-o informatie ce descrie vederea utilizatorului asupra cerintelor. Aceasta informatie ar trebui sa fie suficient de ampla pentru a defini spatiul problemei si pentru obtinerea criteriilor de acceptare a produsului final.

Sintetizarea arhitecturii. Sunt evaluate compromisurile de proiectare, ambiguitatile spatiului problemei si partile disponibile ale spatiului solutiei (tehnologii si componentele existente).

Planificarea si pregatirea unui business case. Sunt evaluate alternativele pentru gestiunea riscurilor, personalului, compromisuri cost/planificare/profitabilitate. Se determina infrastructura (instrumente, procese, suport automatizat) necesara dezvoltarii ciclului de viata.

Criterii principale de evaluare:

- Toti skateholder-ii sunt de acord cu definitia scopului si estimarile de cost si planificare?
- S-au inteles cerintele, sunt acestea puse in evidenta de cazurile de utilizare crtitice?
- Sunt credibile estimarile de cost si planificare, prioritatile, riscurile si procesele de planificare?
- Arhitectura prototip demonstreaza criteriul precedent? (Principalul scop al unei arhitecturi este de a furniza un mijloc pentru intelegerea scopului si atribuirea credibilitatii grupului de dezvoltare in rezolvarea unei probleme tehnice)
- Sunt acceptabile cheltuielile actuale pe resurse vs cheltuielile planificate?

Faza de elaborare

In timpul acestei faze, se construieste o arhitectura prototip executabila, in una sau mai multe iteratii, depinde de scopul, marimea, riscurile si noutatea proiectului. Acest efort se adreseaza cel putin cazurilor de utilizare critice identificate in faza de conceptie, care arata de obicei riscurile cele mai mari.

Obiective principale:

- Stabileste si documenteaza “vederea” externa asupra sistemului
- Stabileste si documenteaza arhitectura sistemului
- Planifica in detaliu faza de constructie

- Se identifica componente ce se pot utiliza in faza de constructie si care ar putea influenta pozitiv asupra predarii

Activitatile fazei de elaborare trebuie sa asigure faptul ca arhitectura, cerintele si planul sunt suficient de stabile si riscurile suficient de moderate, costul si planul pentru terminarea dezvoltarii pot fi prevazute intr-un domeniu acceptabil. Conceptual, acest nivel corespunde necesarului pentru ca firma respectiva sa realizeze faza de constructie la un pret fix.

Activitati principale: (deriva din indeplinirea obiectivelor principale)

- **Elaborarea vederii externe asupra sistemului**. Aceasta activitate implica intelegerea fidela a cazurilor de utilizare critice ce conduc deciziile arhitecturale sau de planificare.
- **Elaborarea arhitecturii sistemului si selectarea componentelor**. Sunt evaluate componentele potentiale si sunt intelese deciziile make/buy astfel incat sa fie determinate planul si costul fazei de constructie.Componentele arhitecturale selectate sunt integrate si atribuite cazurilor de utilizare critice. Aceste activitati pot genera reproiectarea arhitecturii datorita considerarii proiectarilor alternative sau reconsiderarii cerintelor.
- **Planificarea in detaliu a fazei de constructie**. Sunt stabilite procesul de constructie, instrumentele si suportul de automatizare a procesului, milestone-urile intermediare si criteriile lor de evaluare.

Criterii principale de evaluare:

- Vederea externa aspura sistemului este stabila?
- Arhitectura sistemului este stabila?
- Demonstratia executabila arata ca elementele de risc majore au fost rezolvate?
- Faza de constructie este planificata cu suficienta fidelitate?
- Toti skateholder-ii sunt de acord ca daca se respecta planul curent pentru a dezvolta complet sistemul in contextul arhitecturii curente atunci se obtine viziunea (“vederea”) curenta a sistemului?
- Sunt acceptabile cheltuielile actuale pe resurse vs cheltuielile planificate?

Faza de constructie - reprezinta un proces de productie in care accentul cade pe gestiunea resurselor si a operatiilor de control care optimizeaza costurile, planificarile si calitatea (ie performanta).

Obiective principale:

- Detaliaza arhitectura portand-o spre implementare
- Realizeaza iteratii (release-uri) ghidate de riscuri si de cerintele problemei, iteratii controlate de criterii de evaluare reexaminate in mod regulat (optimizarea resurselor, evitarea revenirilor, obtinerea calitatii adecvate)
- Integreaza continuu (produce si testeaza ce produce)

Activitatile principale deriva din indeplinirea scopurilor acestei faze.

Criterii principale de evaluare:

- Aceasta versiune a sistemului este suficient de matura pentru a putea fi data utlizatorilor (in sensul ca defectele existente nu sunt obstacole in atingerea scopului urmatoarei versiuni)
- Aceasta versiune a sistemului este suficient de stabila pentru a fi data utilizatorilor? (in sensul ca modificarile pe loc nu sunt obstacole in atingerea scopului urmatoarei versiuni)
- Toti skateholder-ii sunt pregatiti pentru instalarea produsului la utilizatori?
- Sunt acceptabile cheltuielile actuale pe resurse vs cheltuielile planificate?

Faza de tranzitie-instalare -se ocupa cu activitatile necesare plasarii sistemului in mainile utilizatorilor. De obicei, aceasta faza cuprinde cateva iteratii, inclusiv produse (release) beta, produse valabile si produse imbunatatite. Un effort considerabil este depus pentru dezvoltarea documentatiei utilizatorilor, pregatirea lor si rezolvarea problemelor legate de configurarea, instalarea si utilizarea produsului.

Obiective principale:

- faciliteaza acceptarea clientului (masoara gradul de satisfactie a clientului), imbuntateste prestatiile, elimina erorile)
- porting al bazei de date existente in cea nou, eliminarea graduala a partilor mostenite (legacy)
- pregatirea personalului de operare si intretinere

Activitatile deriva din indeplinirea scopurilor acestei faze.

Criterii principale de evaluare:

- Clientul este satisfacut?
- Sunt acceptabile cheltuielile actuale pe resurse vs cheltuielile planificate?
- Pentru unele proiecte, acest punct final al ciclului de viata poate coincide cu punctul de inceput al unui ciclu de viata pentru urmatoarea versiune a produsului. (adica continua abordarea evolutiva)
- Fiecare din cele 4 faze este formata din una sau mai multe **iteratii** care produce o versiune a sistemului intr-o forma demonstrabila si evaluata cu ajutorul unui set de criterii.

O **iteratie** reprezinta o secventa de activitati pentru care exista un eveniment intermediar bine-definit (milestone minor); scopurile si rezultatele unei iteratii sunt captate cu ajutorul artefactelor discrete. Fiecare iteraţie contine activitatile de analiza, proiectare, implementare şi testare şi produce un sistem incomplet (numit versiune a sistemului final) care este testat, integrat şi executat. Iteratia se intinde pe o perioada fixata de timp, de exemplu de 4 saptamani.

In general, fiecare iteratie rezolvă cateva din cerintele clientilor si a utilizatorilor si extinde incremental sistemul. De exemplu, o iteratie poate pune accentul pe imbunatatirea performantei sistemului. Performanta include cerinte de viteza si spatiu impuse sistemului

Criterii de performanta	Definitie
Timp de raspuns	Cat de repede stie sistemul de cererea utilizatorului dupa ce aceasta a fost transmisa?
Iesirea	Cat de multe task-uri realizeaza sistemul intr-o perioada de timp?
Memoria	De cata memorie are nevoie sistemul pentru a rula?

Fiecare faza corespunde completarii unui numar suficient de iteratii pentru a atinge o anumita stare a proiectului. Tranzitia de la o faza la alta este mai mult o decizie business (de firma) decat terminarea unei activitati specifice a dezvoltarii sistemului.

ARTEFACT-ELE PROCESULUI

Un **artefact** este o colectie coeziva, discreta de informatie, de obicei dezvoltata si revizuita ca o entitate unica (⇔o informatie produsa, modificata sau folosita de un proces).

Colectii distincte de informatii sunt organizate in **seturi de artefacte.** Fiecare set contine artefacte legate, persistente si reprezentate intr-un anumit format (text, model UML, template de document standard, template de foaie de calcul standard, limbaje de programare: C++, Java, Visual Basic).

Artefactele ciclului de viata sunt impartite in 5 seturi dupa formatul limbajului de baza al setului:

- management (format textual adhoc)

- cerinte (text organizat si modele ale spatiului problemei)
- proiectare (modele ale spatiului solutie)
- implementare (limbaj de programare si fisierele susrsa asociate)
- repartitie (limbal procesabil masina si fisierele asociate)

Obs: Ultimele 4 seturi se numesc **seturi ingineresti** de artefacte.

Discipline

Modelul RUP descrie activitatile care au loc in discipline (pana in 2001 se numeau workflow-uri). O disciplina este un set de activitati şi artefactele legate de ele într-o arie de interes sau cu un obiectiv comun.

Activitatile procesului sunt organizate in 7 discipline majore:

- Disciplina managementului: controlul procesului si asigurarea conditiilor de succes tuturor skateholder-ilor
- Disciplina cerintelor: analiza spatiului problemei si construirea artefactelor de cerinte
- Disciplina proiectarii: modelarea solutiei msi construirea artefactelor de arhitectura si proiectare
- Disciplina implementarii: programarea componentelor si construirea artefactelor de implementare si repartitie
- Disciplina evaluarilor: evaluarea tendintelor din proces si a calitatii produsului

S-au omis:

- **documentatia**, deoarece aceasta ar trebui sa fie un produs secundar al celorlalte activitati
- **asigurarea calitatii**, deoarece aceasta este un obiectiv al *tuturor* activitatilor

Artefactele si accentul fiecarei discipline pe fiecare faza a ciclului de viata

Disciplina	Artefacte	Fazele ciclului de viata
Management	Business case	Conceptie: Business case si vederea din exterior a sistemului
	Plan de dezvoltare software	Elaborare: Planul de dezvoltare software
	Avizarile privind starea proiectului	Constructie: Monitorizarea si controlul dezvoltarii
	Vederea din exterior a sistemului	Tranzitie: Monitorizarea si controlul repartitiei
	WBS	
Mediu	Mediu	Conceptie: Definirea mediului de dezvoltare si a infrastructurii de gestiune a modificarilor
	Baza de date a cererilor de modificare	Elaborare: Instalarea mediului de dezvoltare si bazei de date a cererilor de modificare
		Constructie: Intretinerea mediului de dezvoltare si bazei de date si a cererilor de modificare

Disciplina	Artefacte	Fazele ciclului de viata
		Tranzitie: Tranzitia mediului de dezvoltare si a bazei de date a cererilor de modificare
Cerinte	Setul de cerinte	Conceptie: Definirea conceptului operational
	Specificatiile de predare	Elaborare: Definirea obiectivelor arhitecturii
	Vederea din exterior a sistemului	Constructie: Definirea obiectivelor iteratiei
		Tranzitie: Rafinarea obiectivelor predarii (release)
Proiectare	Setul de proiectare	Conceptie: Formularea conceptului de arhitectura
	Descrierea arhitecturii	Elaborare: Obtinerea liniei arhitecturale
		Constructie: Proiectarea componentelor
		Tranzitie: Rafinarea arhitecturii si a componentelor
Implementare	Setul de implementare	Conceptie: Suportul prototipurilor de arhitectura
	Setul de repartitie	Elaborare: Producerea liniilor arhitecturale
		Constructie: Producerea completa a componentelor
		Tranzitie: Intretinerea componentelor
Evaluarea	Specificatiile de predare	Conceptie: Evaluarea planurilor, a vederii din exterior si a prototipurilor
	Descrierile de predare	Elaborare: Evaluarea arhitecturii
	Manualul utilizatorului	Constructie: Evaluarea predarilor temporare
	Setul de repartitie	Tranzitie: Evaluarea produselor obtinute
Repartitie	Setul de repartitie	Conceptie: Analizarea utilizatorilor
		Elaborare: Defineste manualul utilizatorului
		Constructie: Pregatirea materialelor de tranzitie

Disciplina	Artefacte	Fazele ciclului de viata Tranzitie: Transferul produsului la utilizator

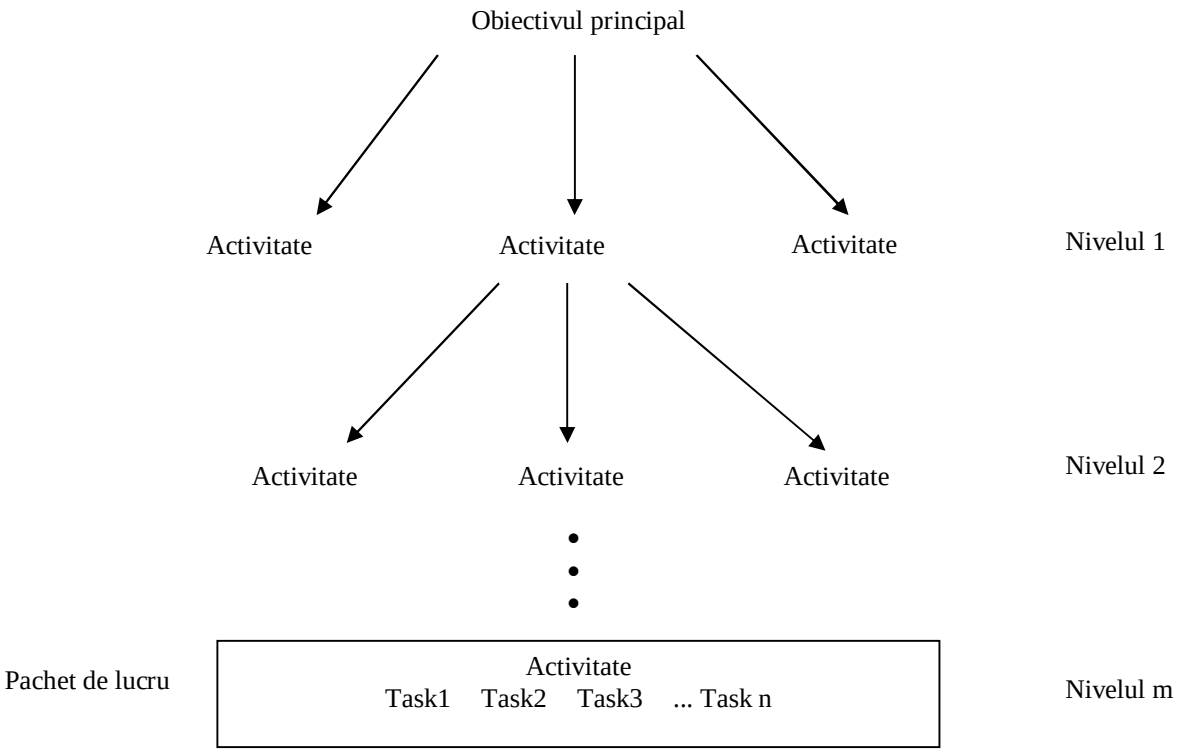
Business case: furnizeaza toate informatia necesara determinarii daca se merita sa se investeasca in proiectul respectiv. El detaliaza venitul asteptat, costul asteptat, planuri tehnice si de management si date backup necesare pentru a demonstra riscurile si realismul planurilor. De obicei un business case contine:

- Contextul (domeniu, piata, scop)
- Abordarea tehnica
- Planul de realizare a caracteristicilor proiectului
- Planul de obtinere a calitatii
- Trade-off-uri ingineresti si riscuri tehnice
- Abordare de management
- Orarul si evaluarea planificarii riscurilor
- Masuri obiective a succesului
- Anexe
- Predictie financiara
- Cost estimativ
- Venit estimativ
- Bazele estimarilor

WBS (Work Breakdown Structure) este o ierarhie a elementelor ce descompun planul proiectului in task-uri discrete. Deci WBS est o structura ierarhica a lucrului ce trebuie facut pentru a termina proiectul asa cum este descris in POS. WBS-ul furnizeaza urmatoarele informatii:

- impartire a activitatilor proiectului
- descompunere clara pentru atribuirea responsabilitatilor
- un framework pentru orar, buget si supravegherea cheltuielilor

Exemplu de WBS:



O activitate este formata din mai multe task-uri. Un pachet de lucru este o descriere completa a task-urilor ce formeaza o activitate care va fi realizata. Pachetul de lucru poate fi foarte simplu sau poate fi un mini-proiect care are toate proprietatile unui proiect oarecare, exceptand faptul ca activitatea ce defineste acest proiect indeplineste cele *6 criterii* si nu trebuie impartita mai departe. Activitatile sunt descrise prin verbe (lucru=>actiune=>verb).

Def. Impartirea muncii intr-o ierarhie de activitati, task-uri si pachete de lucru se numeste **descompunere.**

Aceasta descompunere este importanta deoarece permite estimarea duratei proiectului, determina resursele necesare si planifica munca.

In exp de mai sus, obiectivul principal din POS este definit ca o activitate de nivel 0 in WBS. Nivelul 1 este o descompunere a activitatii de pe nivelul 0 intr-un set de activitati definite ca activitati de nivel 1. Cand fiecare activitate de pe nivelul 1 s-a terminat, rezulta ca activitatea de pe nivelul 0 s-a terminat (in acest caz, rezulta ca proiectul este complet).

Obs. Exista o similitudine intre decompunerea ierarhica si descompunerea functionala. In principiu, nu este nici o diferenta intre un WBS si o descompunere functionala a unui sistem.

WBS-ul este utilizat ca:

Instrument de gandire a procesului. Ajuta managerul proiectului si echipa sa vizualizeze exact cum poate fi definita si condusa munca proiectului.

Instrument de proiectare arhitecturala. WBS este o imagine a lucrului la proiect si cum aceste elemente de lucru sunt legate intre ele. In acest context, WBS devine un instrument de proiectare.

Instrument de planificare. In aceasta faza de planificare, WBS furnizeaza echipei proiectului o reprezentare detaliata a proiectului ca o colectie de activitati ce trebuie terminate pentru ca proiectul sa fie terminat. Ne vom afla la nivelul cel mai de jos al WBS cand vor fi estimate efortul, timpul si necesarul de resurse; se va face un orar ce cuprinde estimarea datelor de furnizare a produselor si terminarea proiectului.

Instrument de raportare a starii proiectului. Activitatile proiectului sunt consolidate de la baza pe masura ce sunt terminate activitatile de pe nivelele inferioare. Unele din activitatile de pe un nivel inferior pot reprezenta un progres semnificativ astfel incat terminarea lor va genera milestone-uri in cursului ciclului de viata al proiectului. Acestea pot fi raportate conducerii superioare si clientului.

WBS-ul procesului modern (iterativ, bazat pe componente –RUP)

organizeaza elementele de planificare in jurul procesului framework si nu al produsului

este format din 2-3 nivele.

WBS-ul ar trebui organizat astfel:

- primul nivel contine *workflow*-urile (management, mediul, cerinte, proiectare, implementare, testare si repartitie). Aceste elemente sunt de obicei atribuite unei singure echipe si constituie anatomia unui proiect pentru planificare si compararea cu alte proiecte.
- elementele celui de-al doilea nivel sunt definite pentru *fiecare faza a ciclului de viata* (conceptie, elaborare, constructie, tranzitie). Aceste elemente permit ca fidelitatea planului sa se dezvolte mult mai natural conform cu nivelul de intelegere a cerintelor si arhitecturii.
- pe nivelul 3 sunt activitatile ce produc *artefactele* fiecarei faze. Aceste elemente pot fi la nivelul cel mai de jos sau pot fi descompuse in activitati de pe alte cateva subnivele, astfel incat, luate impreuna, produc un singur artefact.

Setul de cerinte (Produsele Analizei cerintelor)	*Setul de proiectare (Produsele Proiectarii)*	Setul de implementare (Produsele Implementarii	Setul de repartitie (Produsele Repartitiei)
1. Documentul de "vedere" din exterior al sistemului (documentul de cerinte, diagrama de context) 2. Modelele cerintelor (modelul functional, diagramele de domeniu, prototipuri)	1. Modelele proiectarii (structural, comportamental, dinamic) 2. Modelul testelor 3. Descrierea arhitecturii software	1. Codul sursa item-izat 2. Fisere "make" (script-uri de compilare, fisere .ini, fisiere de date de test si fisiere de date rezultate ale testelor) 3. Componente executabile	1. Produsul executabil integrat 2. Fisiere de run-time asociate 3. Manualul utilizator

Setul de management (Produsele Management-ului)

Produse de planificare	**Produse operative**
1. Work breakdown structure (WBS) = descrierea activitatilor + mecanismul de urmarire financiara a utilizarii bugetului 2. Business case: cost, planificare, profit asteptat 3. Specificatiile de predare (validare): scop, planificare, obiectivele release-ului 4. Planul de dezvoltare a software-ului (SDP)	5. Descrierea predarilor 6. Avizarile privind starea proiectului 7. Baza de date a cererilor de modificare (SCO) 8. Documentele de repartitie (deployment) a software-ului in mediul de operare 9. Mediul (instrumente Hw si Sw,automatizarea PS, documentare, training necesar pentru dezvoltare si productia artefactelor ingineresti)