**Universitatea Politehnica Bucuresti**

**Facultatea de Electronica, Telecomunicatii si Tehnologia Informatiei**

**Tehnici de implementare software**

**Probleme ce pot aparea in implementarea software si rezolvarea acestora**

**Studenti:**

**Benciu Florin**

**Cristina Tudor**

**2014**

**Cuprins:**

1. **Introducere –** Tudor Cristina
2. **Implementarea produsului software –** Tudor Cristina
3. Rolurile care apar in implementarea software
4. Pasii implementarii produsului software
5. Probleme in implementarea software si motivele pentru care apar
6. **Necesitati in implementarea produselor SW complexe –**Benciu Florin
7. Ciclul de viata al produsului software
8. Cresterea complexitatii
9. PDM & SCM (Product data management & Software configuration management)
10. **Tipuri de implemenare software –**Benciu Florin
11. DSDM
12. Prince 2
13. **Metode embedded si generice de implementare software-** Tudor Cristina
14. **Bibliografie**
15. **Introducere**

Metodele de implementare a produselor software este o abordare structurata sistematic pentru a integra efectiv un serviciu sau o componenta bazate pe software intr-un circuit operational dintr-o structura organizationala sau pentru un client final individual.

Programarea produselor software, este procesul care porneste de la formularea problemei de catre client pana la realizarea programului executabil. In ingineria software implementarea este privita ca una din fazele procesului de dezvoltare software.

Implementarea unui produs software pote fi considerata ca un plan de actiune in care utilizatorii sau organizatiile sa poata rula un anumit produs software.

Aceste metode de implementare contin un set de reguli si de vizualizari pentru a putea face fata celor mai comune probleme care pot aparea in timpul implementarii unui produs software: alinierea atat din perspectiva business a organizatiei, cat si din perspectiva umana.

In industria IT, implementarea este procesul de dupa vanzarea produsului, in care clientul este ghidat de la cumpararea produsului.[[1]](#footnote-1)

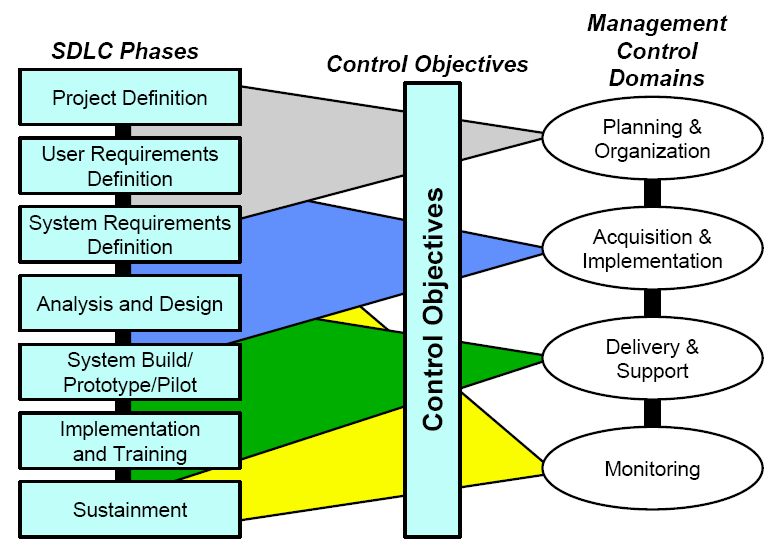


Figura 1[[2]](#footnote-2). Diagrama reprezinta ciclul de viata al unui produs, atat partea tehnica, cat si partea manageriala. In partea dreapta sunt reprezentate fazele ciclului de viata dintr-un sistem de dezvoltare pe parte tehnica. In dreapta sunt reprezentate domeniile din controlul managementului care intervin la fiecare faza din ciclul de viata. Asupra tuturor intervine controlul obiectivelor.

Implementarea produsului software, impreuna cu suportul tehnic, formeaza ultimii pasi din ciclul de viata al unui sistem de dezvoltare software. Implementarea si training-ul au impact financiar foarte puternic.

In general, implementarea unui produs software este aproximativ 1/3 din pretul total al produsului software (chiar mai mult decat cerintele software si hardware la un loc).

In ultimii ani, din ce in ce mai multe firme ofera o implementare si un training advectat pentru produsele pe care le produc. Mai mult, exista si firme de consultanta care se ocupa cu suport tehnic al produselor software. [[3]](#footnote-3)

Sistemul de implementare beneficieaza in general atat de o mare implicare din partea clientului, cat si din partea specialistilor software. Participarea utiliatorului in partea de design si operare a sistemului informatic, are numeroase beneficii. In primul rand, daca utilizatorii sunt foarte implicati in design-ul sistemelor, oportunitatile sunt mai mari de a se creea un produs mulat dupa prioritatile si cerinte lor, deci si usurinta de a controla produsul mai usor. Cu alte cuvinte, cu cat implicarea este mai mare in fazele de concepere a produsului cu atat necesitatea unui suport sau unui training foarte riguros scade. In al doilea rand, utilizatori vor reactiona intr-un mod mai pozitiv la procesul de schimbare. Inglobarea cunostiintelor utilizatorului si a expertilor conduce la solutii mai bune. [[4]](#footnote-4)

1. **IMPLEMENTAREA PRODUSULUI SOFTWARE**
2. **Rolurile care apar in implementarea software**

Complexitatea si varietatea produselor software a condus la specializarea rolurilor pentru coordonarea si rularea procesului de implementare.

De exemplu, pentru sistemele desktop, utilizatorii finali devin de multe ori chiar cei care implementeaza software-ul, atunci cand instaleaza pachetul software pe calculatoarele lor.

Implementarea unui produs software intr-o intreprindere implica mult mai multe roluri. De obicei, aceste orluri se schimba pe masura ce aplicatia progreseaza din faza de testare ( pre-productie) pana la faza de productie.

Rolurile implicate in implementarea produselor software intr-o intreprindere sunt:

* pentru faza de preproductie:

- dezvoltatorii de aplicatie : cunoscutii ca si dezvoltatorii ciclului de viata; ei se ocupa de planificare, implementare codului, testare si documentare, implementare si mentenanta.

- inginerii care construiesc si aduc produsul in faza release (produsul este in versiune functionala) : ei se ocupa de compliare, asamblare si livrarea codului sursa pentru a produs final sau alte componente software. [[5]](#footnote-5)

- managerii care se ocupa de produsul release: se ocupa de managementul inca din faza de dezvoltare pana la terminarea produsului.

- coordonatorii de implementare: se ocupa de coordonarea suportului, training-uluisi mentenantei.

* pentru faza de productie:
* administratorul de sistem: este persoana responsabila de intretinerea si configurarea sistemelor de calculatore. mai ales sistemelor multi-utilizator cum ar fi serverele.
* administratorul de baze de date: acesta este responsabil pentru instalarea, configurarea, imbunatatirea, monitorizarea, mentenanta si securitatea bazelor de date dintr-o organizatie.
* coordonatorii produsului in stadiu release
* managerii de proiect: acestia au responsabilitatea planificarii, executiei si inchiderii unui proiect. [[6]](#footnote-6)

1. **Pasii implementarii produsului software**

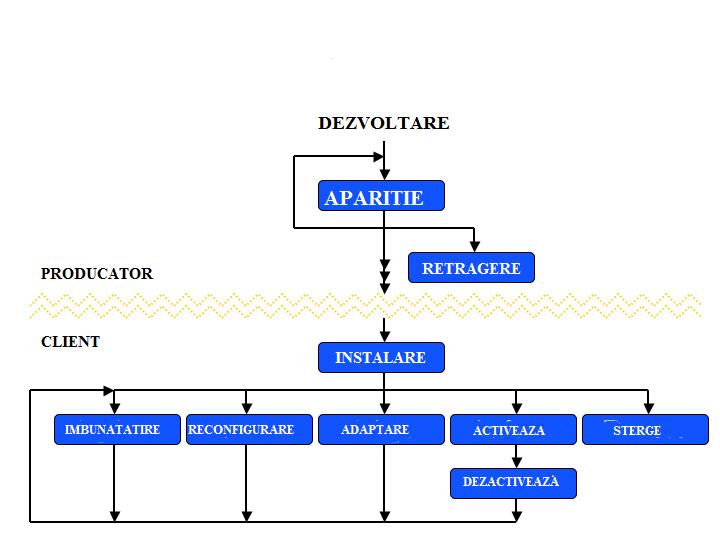


Figura 2[[7]](#footnote-7). Ciclul de viata al implementarii software.

**PASUL 1 – APARITIA**

Ciclul de viata al aparitiei unui produs software este suma tuturor stagiilor de dezvoltare si de maturitate a acelui produs.

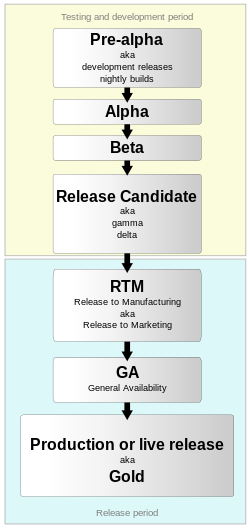


Figura 3[[8]](#footnote-8). Cilul de viata al aparitiei produsului software

Versiunile din ciclul de viata, in ordinea aparitiei, sunt:

* pre-alpha: se refera la toate activitatile desfasurate in timpul proiectului inainte de testare.Toate aceste activitati includ: cerinte de analiza, design de software, dezvoltare software, teste unit. [[9]](#footnote-9)
* alpha: este prima faza din care se incep testarile. De obicei in acesta faza inginerii testeaza doftware-ul folosind metoda “cutiei albe” de testare. Validari aditionale se realizeaza apoi,de catre alte echipe, folosind tehnica “cutiei negre” si cea a “cutiei gri”.
* beta: in general incepe atunci cand toate caracteristicile sunt complete. Software-ul in versiunea beta are multe problemele fata de versiunea completa, la fel si viteza si performanta redusa.
* beta inchisa sau deschisa: varianta beta inchisa apare doar pentru un grup restrans de utiliatori cu scopul testarii; cei care testeaza versiunea beta deschisa fac parte dintr-un grup mai mare, practic, orice este interesat.
* candidatul aparut (RC): este o versiune beta cu potential de a deveni produs final
* general valabil (GA): este produsul in stagiul de marketing, in care toate comercializarile necesare au fost facute iar produsul software este gata de a fi cumparat.
* aparut pentru consum (RTM): este produsul gata de a fi livrat catre client.

[[10]](#footnote-10)

**PASUL 2- INSTALAREA**

Instalarea unui produs software este actul prin care programul este gata de executie. Din cauza variatiei fiecarui program si al fiecarui calculator, produsele software vin la pachet cu un installer, program specializat responsabil de toata instalarea.

Unele programe software pot fi “instalate” doar prin simpla copiere a lor intr-un folder pe calculator, dupa care pot fi executate. Alte programe necesita o procedura mai complexa de instalare.

Operatiile executate in timpul instalarii unui program pot fi:

* asigurarea cerintelor sistemului
* verificarea daca exista versiuni anterioare ale software-ului deja instalate
* crearea sau imbunatatirea fisierelor de program
* adaugarea datelor de configurare cum ar fi fisiere de configurare, intrari de registre Windows sau variabile de mediu
* facilitarea accesibilitatii ultilizatorului la produsul software, creand link-uri, shortcut-uri sau bookmark-uri
* configurarea componentelor care ruleaza automat
* activarea produsului software

Tipuri de instalare:

* instalare care necesita prezenta unui user pentru a face anumite alegeri, cum ar fi acceptarea sau nu a licentei end-user (EULA), specifica preferinte cum ar fi locatia de instalare, parole, sau asistenta in activarea produsului
* instalarea care nu afiseaza mesaje sau ferestre in timpul procesului
* instalare care se efectueaza fara interventia unui user. Acest tip de instalare fie nu necesita informatii din partea user-ului, fie acele informatii au fost setate apriori.
* instalare care nu necesita conectarea unui monitor. In acest tip de instalare alta masina se concteaza la masina target si preia afisarile iesirii.
* instalare care ruleaza la un timp stabilit sau in functie de o conditie[[11]](#footnote-11)
* instalare in absenta oricarui element cu care ar putea interfera, cum ar fi versiuni mai vechi ale software-ului
* instalarea unui program dintre retea su resurse partajate

**PASUL 3- IMBUNATATIRE**

O imbunatatire al unui produs inlocuieste o versiune anterioara cu totul, sau doar o parte din ea, cu o versiune mai noua.

De obicei strategia marilor firme este de a vinde la versiunile imbunatatite cu un discount, astfel clientul folosind in continuare produsele lor. Pentru aceasta, trebuie sa detii o versiune mai veche a programului pentru a beneficia de acel discount. De asemenea, rutina de instalare a versiunilor imbunatatite verifica adesea daca o versiune mai veche este instalata pe acel calculator, altfel instalarea versiunii imbunatatite nu poate avea loc. [[12]](#footnote-12)

**PASUL 4-RECONFIGURARE**

Reconfigurarea unui produs software inseamna adaugarea sau dezactivarea anumitor functionalitati, adaptarea unor functii, sau ajustarea interactiunii software-ului cu mediul extern. Reconfigurarea poate inseamna: schimbarea codului sursa si reconstruirea lui, schimbarea fisierelor de configurare si repornirea aplicatiei, schimbarea dinamica a starii software-ului. [[13]](#footnote-13)

**PASUL 5-ADAPTARE**

Activitatea de adaptare este tot un proces de modificare a produsului software care a fost instalat anterior. Diferenta dintre adaptare si imbunatatire este aceea ca adaptarile sunt initiale de evenimente locale. [[14]](#footnote-14)

**PASUL 6-ACTIVARE**

Activarea este operatia de pornire a componentelor software executabile. Pentru sisteme simple, implica stabilirea unor comenzi pentru executie. Pentru sistemele complexe, tot sistemul de suport ar trebui sa fie pus in functiune.

Din cauza pirateriei crescute si altor forma de folosire neautorizataa produselor software, utilizatorii nu pot fi intotdeauna siguri ca detin un produs original. Scopul activarii unui produs software este diminuarea ratei de piraterie, numita copiere obisnuita. Aceasta copiere este o forma de distribuire si instalare a produselor software care nu o licenta de folosire a produsului. Activarea produsului software ajuta la instalarea acestuia, cu certitudinea ca produsul este original. [[15]](#footnote-15)

**PASUL 7-DEZACTIVAREA**

Este operatia inversa celei de activare, si se refera la inchiderea oricaror componente executabile ale sistemului.

De asemenea, dezactivarea se poate face pe baza unui cod, in functie de sistemul de dezactivare implement la fiecare produs software in parte.[[16]](#footnote-16)

**PASUL 8-STERGERE**

Este pasul opus instalarii, in care se sterge un sistem de care nu mai este nevoie. De asemenea, implica reconfigurarea altor sisteme software pentru a se putea sterge dependentele si unele fisiere ale programului respectiv. [[17]](#footnote-17)

**PASUL 8-RETRAGERE**

Retragerea marcheaza finalul ciclului unui produs software. Produsul este marcat ca si invechit si suportul din partea dezvoltatorilor se retrage. [[18]](#footnote-18)

1. **Probleme in implementarea software si motivele pentru care apar**

Modelarea proceselor, folosita pentru a aduce la acelasi nivel produsul software si structura organizationala, implica o mare dezbatere, atunci cand se ajunge la concluzia ca produsul software si structura organizationala nu sunt la acelasi nivel astfel incat software-ul sa poata fi implementat. In acest caz, exista doua alternative: fie software-ul va fi customizat, fie structura organizationala va fi remodelata, dupa procesele business-ului.

Customizarea unui produs software transforma practic acel produs in unul facut la comanda, astfel ideea de standardizare dispare. Acest lucru poate conduce la pieredea suportului tehnic pentru acel produs. Totusi, customizarea unui produs apare atunci cand integritatea organizationala nu este adaptata, lucru care face ca clientul final sa fie mai putin implicat in acesta etapa, de asemenea sunt necesare mai putine schimburi de tura in fluxul de lucru.

Remodelarea proceselor business-ului poate cauza opunere in folosirea produsului software, deoarecere modificand procesele unui business se vor modifica si task-ruile si responsabilitatile pentru ultilizatorii finali ai produsului software.

Alta problema in implementarea unui produs software este in alegerea extensiei pentru care se vor aplica metode de implementare.[[19]](#footnote-19)

Cateva dintre cele mai comune motive pentru care apar problemele in implementare:

1. Nu au fost definite clar obiectivele proiectului

Nedefinirea clara a obiectivelor implementarii produsului software este un factor care contribuie la nerealizarea corecta a acesteia. Desi definirea obiectivelor este un pas extrem de important, foarte multe firme pornesc un proiect vag, definit partial, cu o tinta nemasurabila.

1. Planificare si management de proiect slab

O planificare slaba si un managent de proiect ineficient nu numai ca va conduce la o implementare proasta, dar poate provoca si pierderea posturile celor care au dat gres.Se intampla de multe ori ca angajatii deja existenti, trebuie sa faca, pe langa munca zilnica, si implementarea noului produs software. Astfel, va exista o perturbatie in procesele zilnice.

1. Adaugarea unor foarte multe trasaturi produsului

O abordare in faze este o tactica mult mai buna: intocmirea unei liste cu factorii care asigura succesul si axarea pe acele trasaturi ale produsului care ajuta la indeplinirea acelori factori. De asemenea, cu cat un produs este mai customizat, cu atat se poate depasi bugetul si termenul limita.

1. Training si suport tehnic insuficient

Lipsa unui training si suport tehnic adecvat in momentele cheie ale proiectului pot duce la insuccesul lui. Unul din motivele principale pentru care un produs nu are succes pe piata este lipsa unui training si suport tehnic inainte si dupa implementare. De multe se intampla acest lucru nu din cauza ca nu exista training sau suport, ci din cauza ca nu sunt facute specific pentru client.

1. Date incomplete, eronate sau fictive in produsul software

Acest este un alt motiv pentru care utilizatorii nu vor cumpara produsul. Atat partea de marketing cat si cea de vanzari nu pot fi efectuate asa cum trebuie cat timp in produs exista inregistrari incorecte.

1. Neplanificare in cazul schimbarilor in produs
2. Platforma hardware si retea instabile pentru suportul produsului software [[20]](#footnote-20)

O platforma hardware si o retea instabila pericliteaza integritatea produsului software. Este o situatie inspecial intalnita in locurile in care conexiunea la internet este foarte slaba.

1. Alegerea gresita a produsului software sau o distribuitorului

Daca distribuitorul de software nu are expertiza in domeniu, nu poate intelege modelul clientului de business sau nu poate training adecvat angajatilor, atunci implementarea nu va avea succes. [[21]](#footnote-21)

1. **Solutii pentru o implementare eficienta a unui produs software**

Proiectul de implementare a unui produs software are anumite trasaturi de care trebuie a tinem cont pentru a realiza un proiect de succes:

* desi exista similaritati intre proiecte, fiecare este diferit de cel precedent. Desi este benefic de a avea in echipa de proiect personal care are experienta in implementarea software, trebuie constientizat faptul ca vor exista provocari in acel proiect
* fiecare proiect are stabilit foarte clar data de inceput si data de final, dar si termene limita intermediare. Livrabilele trebuiesc produse pentru a indeplini un anumit set de obiective. Desi pot aparea modificari in domeniul timpului sau al resurselor, administrarea calendarului proiectului este critic pentru succes.
* conducerea aproba si aloca un anumit buget pentru fiecare proiect. El trebuie respectat pentru a indeplini cerintele clientului
* alocarea resurselor (efortul depus, echipamente,materiale, oameni) trebuie sa fie foarte clar specificata de la inceput
* orice proiect are un grad de risc si de incertitudine, care trebuie sa fie luat in calcul

1. **Planificarea proiectului**

O buna parte din timpul planificarii proiectului este alocata analizarii obiectivelor si tinte, atat din perspectiva personala, cat si functionala.

Obiectivele ar trebui definite in termeni de business, fara a face referire la calculatoare sau produse software. Aceasta permite o incadrare mai buna a prioritatilor si masurarii sucessului.

Atunci cand problema este cuantificata in termeni de business, ceea ce rezulta poate fi folosit ca si punct de referinta prin care se paote masura succesul.

De asemenea, si obiectivele pot ajuta la o incadrare a lucrurilor impartante pentru a face o implementare buna. [[22]](#footnote-22)

1. **Management de top**

Managementul trebuie sa fie adaptat la schimbarile procedurale si culturale, care reprezinta cheia spre beneficiile reale ale schimbarii unui sistem.

Un management de calitate determina succesul real al unui proiect.

1. **Analiza “golurilor”**

Se analizeaza procesele actuale ale companiei versus procesele ce se doresc a fi rulate, si se urmareste daca intre cele doua exista diferente. Cu siguranta noul produs software va manevra altfel procesele, si nu va avea nevoie de toate functionalitatile proceselor ca vesiunea anterioara.

1. **Schimbarea procedurilor**

Adeseori in proiect pot aparea schimbari. Important este ca fiecare schimbare sa fie foarte bine documentata, sa se inteleaga de ce s-a produs, care este impactul asupra costului si calendarului proiectului, cine va fi responsabil pentru monitorizarea schimbarilor, si cine este responsabil pentru costurile produse.

1. **Roluri si responsabilitati**

In echipa de realizare a unui proiect de implementare software trebuie introdusi oameni care inteleg procesele companiei.

Primul criteriu in alegerea membrilor este ca ei sa aibe o atitudine pozitiva, mai ales in cazul rolului de conducator. Schimbarilor pot fi dificile, de aceea o atitudine pozitiva poate scoate echipa din impas.

O echipa de implementare ar trebui sa aibe oameni care sa indeplineasca urmatoareleroluri:

* sponsor executiv (persoana care este dedicata succesului aceluiproiect si are autoritatea sa ia deciziile cele mai grele sin situatiile critice)
* managerul de proiect (balanseaza si manevreaza calendarul proiectului, resursele, si bugetul)
* persoane care exceleaza pentru fiecare zona functionala din companie
* personal de vanzari
* consultanti[[23]](#footnote-23)

1. **Bugetul**

Pentru o stabilire a bugetului cat mai exacta, trebuie ca echipa de management sa fie realistica si sa include atat costurile interne cat si cele externe. Pentru multe companii este tentant sa se scada din bugetul alocat pentru partea hardware sau pentru implementare. Tehnica Lean Thinking – Six Sigma propune mai intai evaluarea proceselor care aduc valoare si celor care nu aduc valuare si diminuare pe cat posibil a celor din urma.

1. **Implicarea din partea sustinatorilor proiectului**

Pe masura ce planul de implementare este realizat, trebuie primita din partea sustinatorilor aprobare. Rezistenta naturala la schimbare este diminuata daca oamenii se simt implicati in proiect.

1. **Comunicarea**

Cea mai buna solutie de a indeparta frica, incertitudinea, si dubiile personalului este prin intermediul unei comunicare eficiente stabilite la intervale regulate de-a lungul ciclului de viata al proiectului.

Intalnirile strategice includ de obicei implicarea intregii echipe si are ca scop informarea tutoror membrilor echipei despre progresele proiectului, schimbari, probleme aparute, etc.

Intalnirile tacticale sunt destinate anumitor membrii ai echipei pentru a rezolva probleme specifice

1. **Formare profesionala a utilizatorilor**

Utilizatorii care au formare profesionala asupra prodului :

* vor folosi produsul software mai eficient
* vor fi scutiti de frustrarea de a nu intelege de ce produsul nu functioneaza cum trebuie
* vor face mai putine erori

1. **Suport tehnic pentru clienti**

Compania trebuie sa ofere asistenta tehnica pentru produsele software pe care le ofera clientilor lor. Suportul tehnic poate fi oferit fie prin telefon, fie prin e-mail, sau prin intermediul unui site sau forum.

Ideea din spatele asistentei tehnice este ca un client plateste atat pentru produsul **[[24]](#footnote-24)**

1. **Necesități în dezvoltarea produselor SW complexe**
2. **Ciclul de viață al produsului** = ansamblul etapelor parcurse în dezvoltarea unui

produs software

Fazele ciclului de viață:

*Definiția* începe când este formulată problema de rezolvat. Se pune accentul pe CE face programul, și anume:

▫**CE** informaţie se prelucrează?

▫**CE** funcţii sau performanţe trebuie să aibă sistemul?

▫**CE** interfeţe cu alte sisteme?

▫**CE** restricţii de proiectare există?

▫**CE** criterii de validare sunt necesare?

Analiza de sistem (ingineria de sistem) implică rolul pe care-l joacă aplicația în ansamblul organizaţiei, planificarea proiectului, analiza riscurilor, estimarea costurilor, alocarea resurselor necesare pentru dezvoltare, sarcinilor de lucru şi a orarului, analiza cerinţelor, detaliată a informaţiei, specificarea funcţiilor pe care trebuie să le execute aplicația, precizarea restricţiilor impuse asupra aplicației

*Dezvoltarea* pune accentul pe CUM trebuie realizată aplicația. Se defines structurile de

date, arhitectura programului, detalii de implementare a procedurilor şi datelor și testarea.

Mai pe larg, dezvoltarea implică urmutoarele: cerinţele aplicației cu set de reprezentări (grafice, tabelare, bazate pe limbaje de descriere); specificaţii de proiectare cu structuri de date, arhitectura aplicatiei, algoritmi pentru prelucrări ăi caracteristiciale interfeţei cu utilizatorul; dezvoltarea efectivă (coding) impreună cu traduce specificaţiile de proiectare în cod sursă, testare unitară(unit testing) și în final testarea de integrare și de acceptare

*Exploatarea* implică instalarea și întreţinerea produsului software.

*Configurația aplicației* include toate componentele sistemului, documentaţia proiectului şi produsului, codul sursă, datele și programul executabil.

Două lucruri sunt cruciale în functionarea corectă a unui produs software: managementul comlexitășii sale și aprovizionarea de suprot pe toată durata da viață a produsului. Cu cât un produs este mai complex, cu atât procedurile de dezvoltare și support sunt mai complexe.[[25]](#footnote-25)

[[26]](#footnote-26)

Figură: HW și SW costurilor de dezvolatare în industria de roboți

1. **Cresterea complexității**

Multe produse nu functionează după așteptările nostre, dar mai apar și problem grave lugate de proasta funcționalitate (bug-uri, crash-uri, încărări de memorie). Aceste nereguli apar din cauza creșterii complexității programelor software ceea ce implică o scădere a robusteții lor.

Un produs poate fi complex din multe motive, unele dintre aceste motive sunt evidente, altele nu prea. Un produs poate fi complex din punct de vedere software (system de operare), dar ce face ca, construcția unui pod sau dezvoltarea unui produs software să fie complex?

Există niste parametriii care, oarecum, influențează complexitatea produselorȘ

* Complexitatea funcționalității. Există produse ale căror proprietăți cum ar fi încrederea, robustețea, valabilitatea și performanța au o importanță crucială. Aceste tipuri de cerințe nu specific funcționalități particulare, ci un comportamen al produsului. Din aceste motive, aceste tipuri de cerințe sunt implementate ca fiind cereri nonfuncționale sau extra-funcționale, uneori și calitatea serviciilor.Exemple de cerințe non-funcționale care se referă la procesul de dezvoltare: modifiabilitatea, testabilatea, reuzibilitatea și urmărirea produsului. O mulțime de cerințe și o un număr mare de tipuri diferite de cerințe, des incompatibile, duc la multe dificutăți în procesul de dezvoltare.
* Complexitatea structurii: Pentru a evita aceste dificultăți in rezolvarea problemelor complexe, un binecunoscut principiu, *divide et impera*, este des folosit. Prin acest principiu, o problema majoră este divizată in problem mai mici care sunt mai ușor și mai rapid de rezolvat.
* Complexitatea comportamentului operational. În unele cazuri interacțiunile dintre un produs și consumatori pot fi complexe, ceea ce face dificilă predicția comportamentală a produsului în astfel de operaționi. Este cunoscut faptul că sistemele pot avea o funcționalitate superioară legată de operațiuni pentru care produsul nu a fot niciodată proiectat. Cientul poate vrea să folosească produsul într-un mod particular sau pentru un scop particular care necesită remodificarea produsului. Acest lucru, la rândul lui, necesită service, mentenanță sau o nouă dezvoltare a produsului.
* Complexitatea procesului de dezvoltare. Când produsele sunt complexe, atunci procesul de dezvoltare tinde sa fie și el la fel de complicat. Când mulți oameni cu roluri diferite sunt implicați în procesul de dezvoltare, este important pentru ei să aibă acces la informație acută. Când doi oameni comunică direct, unul cu celălalt, există o legătură între cei doi; când sunt trei, exită trei legături, și pentur patru oameni, numărul de legături crește la șase. Pentru un grup de n oameni, sunt create n(n-1)/2 legături direce comunicaționale.
* Dezvolare și producție distribuită. Dezvoltarea și producția pot fi distribuite geographic, amândouă. Pentur produse software, developerii sunt de obicei dispersați

Figură: Relații directe intre membrii unui grup

* pe câteva site-uri, din diferite țări, în timp ce dezvoltă același system software.
* Nevoia de mentenanță. Cererea pentru mentenanță și dezvoltare viitoare implică faptul că trebuie să știm toate status-urile produselor-câteodată trebuie să ținem evidența legată de modificările făcute de client.[[27]](#footnote-27)

1. **PDM & SCM (Product data management & Software configuration management**

PDM-Managementul datelor despre produs SCM-MAnagementul configurației software

Scopul principal al sistemelor PDM este managementul și controlul structurilor mari și complexe de date ale sistemelor CAD/CAM/CAE/CAx.  În anul 1999, sistemele principale PDM includeau funcționalități pentru managementul de documente, managementul configurației/BOM, managementul și controlul fluxului de lucrări de bază. Actualmente, există un consens că PDM este concentrat pe managementul datelor CAx  (tehnologii asistate de calculator pentru proiectare și fabricație).

Sistemul PDM este o subcategorie esențială a sistemului mai larg al PLM și în mod tipic constituie numai componenta de manipulare a datelor în PLM.

Sistemul PDM include următoarele elemente de bază ale soluțiilor PLM:

* + Managementul depozitelor de date și managementul documentelor;
  + Managementul structurii produsului;
  + Managementul proceselor, utilizând fluxuri de lucrări;
  + Managementul clasificărilor (pieselor, componentelor, pe baza unor scheme de clasificare, atribute, coduri etc.)
  + Managementul programelor și proiectelor.

Diferite persoane cu diferite roluri sunt angajate în dezvoltarea sau utilizarea unui produs. Aceste persoane sunt adesea interesate de anumite date despre produs, care sunt importante pentru o fază specifică sau pentru un anumit proces în care sunt implicate. Exemple de astfel de date sunt documente cerute ce sunt create în fazele de dezvoltare timpurii ale produsului, dar şi utilizate în timpul etapelor de testare pentru a verifica dacă funcţionarea produsului şi pentru a valida produsul. Comune pentru toate datele, indiferent de fază în care sunt create, este că ea aparţine produsului. Mai precis specificate, ar trebui să fie legate de produs sau, chiar mai bine, de partea corectă a produsului. Iar pentru un anumit produs, ar trebui să fie posibil să se găsească toate datele relevante pentru acesta.

Domeniul care gestionează aceste date ale produsului în timpul ciclului de viaţă al produsului este numit PDM*. PDM este o disciplină inginerească, care include diverse metode, standarde şi instrumente*. În primul rând, gestionează date referitoare la produse: structuri de produse, inclusiv liste de componente şi configuraţii de produs care identifică toate artifactele (versiunile componente şi documente) aparţinând unei anumite versiuni a produsului. În al doilea rând, aceasta susţine proceduri în timpul PLC (de exemplu, funcţiile oamenilor în procese, inclusiv autorităţile şi alocarea activităţilor lor, managementul schimbării, managementul fluxului de lucru şi chiar şi management de proiect). PDM, se ocupă şi cu dezvoltarea şi producţia de infrastructură, ceea ce înseamnă că acesta oferă informaţii pentru toate activităţile. Acest lucru înseamnă, de exemplu, că PDM nu include metode de proiectare, dar face posibilă disponibilitatea tuturor informaţiilor necesare pentru un proces de proiectare de succes. În mod tradiţional, PDM se ocupă cu produse hardware deşi de obicei nu a fost folosit în dezvoltarea de produse software.

Aşa cum cititorul înţelege, PDM şi SCM trebuie să aibă multe cerinţe şi o funcţionalitate ridicată în comun. De asemenea, chiar dacă aceste zone au o istorie diferită, ambele sunt în creştere şi încorporează din ce în ce mai mult din nevoia totală a gestionării datelor de produs, care rezultă într-o suprapunere semnificativă. Această suprapunere este acum extinsă prin faptul că unii oameni susţin că au nevoie de doar una dintre ele, alegerea fiind determinată dacă software-ul sau hardware-ul este dezvoltat. Există însă diferenţe provenite din natura artefactelor şi diferenţe în procese. Produsele hardware sunt vizibile ca şi componentele lor.

Produsele software nu sunt vizibile în acelaşi mod ca produsele hardware. Acestea sunt încorporate în hardware specifice (scop special calculatoare) sau în hardware standard (calculatoare desktop comune). Structura sistemelor software-ul în timpul de proiectare este de multe ori foarte diferit de structura sistemului în timpul execuției. Deoarece faza de producţie de produse hardware este extrem de importantă şi exigentă, punctul central al PDM pe fază de producţie este foarte mare. Pentru produsele software, producţia este mult mai simplă decât faza de dezvoltare. Procesul de implementare pentru sistemele software este extrem de dinamic, de aceea SCM se concentrează mai ales pe fază de implementare.

Atâta timp cât procesele de dezvoltare hardware şi software sunt izolate una de alta, putem accepta că PDM şi SCM, deşi similare în principiu sunt diferite atunci când sunt puse în aplicare. Se foloseşte un instrument sau altul în funcţie de procesul de dezvoltare şi de producţie avut. Adevărata problemă începe atunci când începe producerea produselor care constau atât din hardware cât şi din software. Întrebarea este dacă costurile pot fi reduse utilizând un instrument în loc de două. O alta întrebarea s-ar lega de modul în care se poate realiza un suport integrat în timpul PLC indiferent dacă se dezvoltă software sau hardware în părţi.[[28]](#footnote-28)

1. **Tipuri de implementare software**

**1. DSDM** - Dynamic Systems Development Method

Metoda DSDM (Dynamic Systems Development Method) este singura metodă din domeniul public pentru RAD (Rapid Application Development), documentată și promovată. Metoda este promovată de Consorțiul DSDM, o organizație non-profit dedicată definirii, promovării si dezvoltării standardului DSDM (http://www.dsdm.org/) Metoda (considerată standard în UK pentru RAD) a fost definită prin evaluarea experienței unui mare număr de consorții de producători (furnizori) și organizații ale utilizatorilor (clienților).

*Principiile de bază*

1.*Implicarea activă a utilizatorului este imperativă (obligatorie)*

DSDM reprezinta o abordare centrata pe utilizator. Daca utilizatorii nu sint implicati puternic in intrgul ciclu de dezvoltare, deciziile pe care va trebui sa le lua vor fi intirziate iar utilizatorul va avea impresia ca solutia finala ii este impusa de proiectanti (dezvoltatori) sau de echipa lor de management. Utilizatorii nu trebuie lasati in afara echipei de dezvoltare ci inauntru, actionind nu numai ca furnizori de informatie si evaluatori ai rezultatelor ci ca participanti activi in procesul de dezvoltare.

2.*Echipele DSDM trebuie sa fie imputernicite sa ia decizii.*

Echipele DSDM constau atit din dezvoltatori (proiectanti) cit si din utilizatori. Ei trebuie sa aiba libertatea (si sa fie in stare) sa ia decizii pe masura ce sint rafinate cerintele si acest lucru solicita modificari. Ei trebuie sa fie imputerniciti sa accepte de pilda ca anumite functionalitati cerute de utilizatori sint acceptabile, fara ca pentru aceasta sa ceara aprobarea nivelelor superioare de management.

3.*Concentrarea este pe livrarea cit mai frecventa a produselor*

O abordare bazata pe produs este mai flexibila decit una bazata pe activitate. Munca unei echipe DSDM este concentrata pe produse care pot fi livrate intr-o perioada de dimp dinainte stabilita. Aceasta abordare va determina echipa sa aleaga cea mai buna cale pentru a obtine produsul (produsele) in intervalul de timp avut la dispozitie. Deoarece activitatea va fi desfasurata in etape scurte, echipa va trebui sa decida care sint activitatile necesare pentru a realiza produsul.

4.*Cel mai importrant criteriu pe baza caruia un produs livrat va fi acceptat este masura in care acesta raspunde cerintelor de afaceri.*

Elementul central al metodei DSDM consta in faptul ca va conduce la livrarea unor produse cu functionalitatea ceruta, la momentul de timp convenit cu utilizatorul. Optimizarile pot fi facute la o data ulterioara.Concentrarea pe respectarea fidela a cerintelor documentelor si pe compatibilitati cu produse anterioare nu se justifica: cerintele pot fi neclare sau se vor modifica, compatibilitatile pot sa nu fie cerute.

5.*Pentru a obtine o solutie potrivita pentru business, trebuie folosita o abordare iterativa si incrementala.*

O dezvoltare incrementala (permisa de DSDM) permite proiectantilor sa utilizeze la maximum feedback-ul de la utilizatori. Mai mult, se pot oferi solutii partiale care sa satisfaca nevoile immediate ale utilizatorului.

6.*Toate schimbarile in timpul procesului de dezvoltare sint reversibile*

Pentru a controla evolutia tuturor produselor (documente, soft, rapoarte de test, etc), starea tuturor acestora trebuie sa fie cunoscuta in fiecare moment (responsabilitatea echipei de management)

7.*Cerintele sint stabilite si "inghetate" la cel mai inalt nivel.*

Principiul ne asigura ca s-a convenit la cel mai inalt nivel scopul si functionalitatile sistemului pentru ca energia sa fie concentrata pe cercetarea implicatiilor cerintelor. O detaliere a cerintelor poate fi convenita mai tirziu, in cursul procesului de dezvoltare dar detalierea nu trebuie sa schimbe semnificativ cerintele initiale.

8.*Testarea este integrata in intrgul proces de dezvoltare*

Cu alte cuvinte, testarea nu trebuie tratata ca o activitate separata. Daca sistemul este dezvoltat incremental, testarea trebuie realizata de asemenea incremental si realizata atit de proiectanti cit si de utilizatori pentru a ne asigura ca procesul de dezvoltare se indreapta in directia buna.[[29]](#footnote-29)

Timp

Resurse

Funcționalitate

Fixat

DSDM

DSDM

Variabil

Tradițional

Resurse

Funcționalitate

Timp

Comparatie intre metodele RAD si cele traditionale.

**2. Prince 2**

Este o metodologie de management de proiect creata de catre guvernul Marii Britanii. Acesta este folosita și în sectorul privat devenind populară în multe țări europene. Este o marcă înregistrata a Office of Government Commerce, și este de asemenea înregistrata la U.S. Patent and Trademark Office.

Caracteristici Prince 2:

|  |
| --- |
| * se descompune in opt procese |
| * Trei procese (Demararea Proiectului, Initierea Proiectului si Planificarea) se refera la planificarea proiectului. Atat Demararea cat si Initierea Proiectului apar la inceputul metodologiei PRINCE2®, iar Planificarea este un proces ce se desfasoara continuu pe parcursul intregii durate a proiectului. * aloca executiei proiectului trei procese. |
| * procesul de Conducere a Proiectului este executat de catre Comitetul Proiectului si nu are legatura cu activitatile zilnice ale managerului de proiect. |
| * Include Ciclul de Viata al Proiectului |
| * Procesul de Finantare a Proiectului este Inclus[[30]](#footnote-30) |

**V. Metode embedded si generice de implementare software**

O metode de implementare software este prin intermediul metodelor sau modelelor embedded. Modelele embedded fac parte din materialele auxiliare care vin impreuna cu pachetul produsului software.

Implementarea unui produs software folosind metodele embedded implica nu numai faptul ca aceste metode sunt folosite numai pentru produsul software respectiv, dar si faptul ca acel produs poate sau trebuie sa fie implementat prin intermediu acelui model. Aceste metode, totusi, pot fi vazute ca niste metode foarte specifice de implementare a unui produs software.

Exemple de produse software care folosesc metode embedded sunt :

* implementarea SAP R/3 (software de planificarea de resurse produs de SAP AG ) folosind modelul embedded ARIS
* implementarea sistemului Baan ERP, folosind Modelarea Dinamica a Intreprinderii (DEM)
* implementarea suitei Oracle E-Business, folosind metoda de implementare Oracle Application (AIM)

AIM este o metoda in 6 faze:

* definirea: in aceasta faza este panificat proiectul, se determina obiectivele, se verifica fezabilitatea proiectului
* analiza operatiilor: include cerintele documentelor, golurile din software, cerintele arhitecturii sistemului
* planificarea solutiilor: se proiecteaza olutiile pentru a indeplini cerintele procesului
* construirea: codarea si testarea customizarilor, interfetele si datele sunt proiectate
* tranzitia: solutia finala este livrata catre intreprindere
* productie: utilizatorii si echipa de implementare retuseaza produsul pentru a minimiza impactul un incidende nefavorabile

Metodele generice de implementare nu sunt create pentru un produs software

specific, ci pentru implementarea generala a produselor software.

Un exemplu de acest tip este implementarea produselor software folosind

metodologia procesului obiect (OPM). OPM este o abordare prin care sistemul informatic este impartit si planificat folosind modele de obiecte si modele de procese.[[31]](#footnote-31)

1. **Bibliografie**
2. <http://en.wikipedia.org/wiki/Product_software_implementation_metho>
3. http://en.wikipedia.org/wiki/Systems\_development\_life-cycle
4. Richard L. MacInnes, “The lean Enterprise Memory Jogger”, Net Resuls Inc., First Edition GOAL/QPC, chapter 4
5. http://en.wikipedia.org/wiki/Software\_deployment#Deployment\_roles
6. http://en.wikipedia.org/wiki/Software\_developmenthttp://en.wikipedia.org/wiki/Software\_deployment#Deployment\_roles
7. http://en.wikipedia.org/wiki/Software\_release\_life\_cycle
8. http://en.wikipedia.org/wiki/Installation\_(computer\_programs)
9. http://en.wikipedia.org/wiki/Upgrade, http://www.glarysoft.com/software-update/
10. http://financialsoft.about.com/od/softwaretitle1/f/upgradevupdate.htm
11. http://en.wikipedia.org/wiki/Software\_configuration\_management
12. http://www.webopedia.com/TERM/C/configuration.html
13. http://en.wikipedia.org/wiki/Software\_Updater
14. http://en.wikipedia.org/wiki/Product\_activation
15. https://jolly.zendesk.com/entries/21998401-Deactivate-your-software-license
16. http://en.wikipedia.org/wiki/Uninstaller
17. http://en.wikipedia.org/wiki/Obsolescence
18. Dana Ginn, Barbara Streibel, “ The design for six sgma Memory jogger”, http://www.softwareceo.com/downloads/file/sceo/white\_papers/SoftwareImplement.pdf
19. http://searchitchannel.techtarget.com/feature/Top-10-pitfalls-when-implementing-Software-as-a-Service
20. http://searchitchannel.techtarget.com/feature/Common-SaaS-problems-that-occur-after-implementation
21. http://www.technologyevaluation.com/research/article/5-Best-Practices-for-Ensuring-a-Smooth-Software-Implementation.html http://www.penta.com/documents/Penta\_Article\_CFMA\_Software\_Implementation\_Best\_Practices.pdf, http://www.crmsearch.com/crm-implementation.php
22. http://en.wikipedia.org/wiki/Product\_software\_implementation\_method
23. http://en.wikipedia.org/wiki/Embedded\_software
24. <http://www.codeproject.com/Articles/5097/What-Is-DSDM>
25. <http://en.wikipedia.org/wiki/Product_software_implementation_method#DSDM>
26. [http://en.wikipedia.org/wiki/Product\_software\_implementation\_method](http://en.wikipedia.org/wiki/Product_software_implementation_method#Prince2)
27. <http://cs.upm.ro/_users/cursuri_on_line/CD/IP/PARADIGMELE/PARADIGMELEIP.HTML>
28. <https://tehnologiidimitrieleonida.wikispaces.com/file/view/T+2_Metodologii+de+proiect.ppsx>

1. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Product\_software\_implementation\_method [↑](#footnote-ref-1)
2. Poza luata de pe pagina http://en.wikipedia.org/wiki/Systems\_development\_life-cycle [↑](#footnote-ref-2)
3. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Product\_software\_implementation\_method [↑](#footnote-ref-3)
4. Richard L. MacInnes, “The lean Enterprise Memory Jogger”, Net Resuls Inc., First Edition GOAL/QPC, chapter 4 [↑](#footnote-ref-4)
5. Informatie de pe pagina http://en.wikipedia.org/wiki/Software\_deployment#Deployment\_roles [↑](#footnote-ref-5)
6. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Software\_deployment#Deployment\_roles [↑](#footnote-ref-6)
7. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Software\_development [↑](#footnote-ref-7)
8. Poza luata de pe pagina http://en.wikipedia.org/wiki/Software\_release\_life\_cycle [↑](#footnote-ref-8)
9. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Software\_release\_life\_cycle [↑](#footnote-ref-9)
10. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Software\_release\_life\_cycle [↑](#footnote-ref-10)
11. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Installation\_(computer\_programs) [↑](#footnote-ref-11)
12. Informatie luata de pe http://en.wikipedia.org/wiki/Upgrade, http://www.glarysoft.com/software-update/, http://financialsoft.about.com/od/softwaretitle1/f/upgradevupdate.htm [↑](#footnote-ref-12)
13. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Software\_configuration\_management, http://www.webopedia.com/TERM/C/configuration.html [↑](#footnote-ref-13)
14. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Software\_Updater [↑](#footnote-ref-14)
15. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Product\_activation [↑](#footnote-ref-15)
16. Informatie luata de pe pagina https://jolly.zendesk.com/entries/21998401-Deactivate-your-software-license [↑](#footnote-ref-16)
17. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Uninstaller [↑](#footnote-ref-17)
18. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Obsolescence [↑](#footnote-ref-18)
19. Richard L. MacInnes, “The lean Enterprise Memory Jogger”, Net Resuls Inc., First Edition GOAL/QPC, chapter4, [Dana](http://www.betterworldbooks.com/list.aspx?SearchTerm=Ginn%2c+Dana) Ginn, Barbara Streibel, “ The design for six sgma Memory jogger”, http://www.softwareceo.com/downloads/file/sceo/white\_papers/SoftwareImplement.pdf [↑](#footnote-ref-19)
20. Richard L. MacInnes, “The lean Enterprise Memory Jogger”, Net Resuls Inc., First Edition GOAL/QPC, chapter4, [Dana](http://www.betterworldbooks.com/list.aspx?SearchTerm=Ginn%2c+Dana) Ginn, Barbara Streibel, “ The design for six sgma Memory jogger”, http://www.softwareceo.com/downloads/file/sceo/white\_papers/SoftwareImplement.pdf [↑](#footnote-ref-20)
21. Informatie luata de pe pagina http://searchitchannel.techtarget.com/feature/Top-10-pitfalls-when-implementing-Software-as-a-Service, http://searchitchannel.techtarget.com/feature/Common-SaaS-problems-that-occur-after-implementation [↑](#footnote-ref-21)
22. Richard L. MacInnes, “The lean Enterprise Memory Jogger”, Net Resuls Inc., First Edition GOAL/QPC, chapter4, [Dana](http://www.betterworldbooks.com/list.aspx?SearchTerm=Ginn%2c+Dana) Ginn, Barbara Streibel, “ The design for six sgma Memory jogger”, http://www.softwareceo.com/downloads/file/sceo/white\_papers/SoftwareImplement.pdf [↑](#footnote-ref-22)
23. Informatie luata de pe http://www.technologyevaluation.com/research/article/5-Best-Practices-for-Ensuring-a-Smooth-Software-Implementation.html, http://www.penta.com/documents/Penta\_Article\_CFMA\_Software\_Implementation\_Best\_Practices.pdf, http://www.crmsearch.com/crm-implementation.php [↑](#footnote-ref-23)
24. Informatie luata de pe http://www.technologyevaluation.com/research/article/5-Best-Practices-for-Ensuring-a-Smooth-Software-Implementation.html, http://www.penta.com/documents/Penta\_Article\_CFMA\_Software\_Implementation\_Best\_Practices.pdf, http://www.crmsearch.com/crm-implementation.php [↑](#footnote-ref-24)
25. Informații luate de pe : http://andrei.clubcisco.ro/cursuri/f/f-sym/4mps/elearn/30%20Ciclul%20de%20viata%20al%20unui%20produs%20software.pdf [↑](#footnote-ref-25)
26. Poză luată de pe : http://books.google.ro/books?id=EAUbUpxf8jMC&printsec=frontcover&dq=Product+software&hl=ro&sa=X&ei=FqLWUtvgNYSihgebhoDYCA&ved=0CDMQ6wEwAA#v=onepage&q&f=true [↑](#footnote-ref-26)
27. Informații luate din : http://books.google.ro/books?id=EAUbUpxf8jMC&printsec=frontcover&dq=Product+software&hl=ro&sa=X&ei=FqLWUtvgNYSihgebhoDYCA&ved=0CDMQ6wEwAA#v=onepage&q&f=true [↑](#footnote-ref-27)
28. Informații luate din : http://books.google.ro/books?id=EAUbUpxf8jMC&printsec=frontcover&dq=Product+software&hl=ro&sa=X&ei=FqLWUtvgNYSihgebhoDYCA&ved=0CDMQ6wEwAA#v=onepage&q&f=true [↑](#footnote-ref-28)
29. Informații de pe : http://cs.upm.ro/\_users/cursuri\_on\_line/CD/IP/PARADIGMELE/PARADIGMELEIP.HTML [↑](#footnote-ref-29)
30. <https://tehnologiidimitrieleonida.wikispaces.com/file/view/T+2_Metodologii+de+proiect.ppsx> [↑](#footnote-ref-30)
31. Informatie luata de pe pagina http://en.wikipedia.org/wiki/Product\_software\_implementation\_method, http://en.wikipedia.org/wiki/Embedded\_software [↑](#footnote-ref-31)