|  |
| --- |
| Universitatea Politehnica București |
| Ciclul de viață al unui produs software |
| Proiec IS |
|  |
| **Gheață Ioana Cristina, 442A** |
| **2/2/2015** |

Cuprins

**Capitolul 1.**

**Ce reprezintă ciclul de viață al unui produs Software?**

**Capitolul 2.**

**Logica înlănțuirii fazelor unui ciclu de viață**

**Capitolul 3.**

**Modeluri de cicluri de viață și alegerea unui ciclu de viață potrivit**

* **Modelul cascadă**
* **Modelul în V**
* **Modelul iterativ**
* **Agille programming**
* **Modelul spirală**

**Capitolul 4.**

**Concluzii**

**Capitolul 1.**

**Ce reprezintă ciclul de viață al unui produs Software?**

Un ciclu de viață al unui produs software reprezintă un set de activități care conduc spre producearea unui produs software. Aceste activiăți implica dezvoltarea software de la zero într-un limbaj conceptual precum Java sau C. În zilele noastre, din ce în ce mai multe produse software sunt dezvoltate prin îmbunătățire unor produse deja existente sau plecând de la un anumit model.

Procesele de creare software sunt complexe și se bazează pe decizile si judecata unor oameni. Datorită necesității de originalitate, luare de deciziii, modelare pe anumite modele, automatizarea acestor procese este foarte limitată. Nu exista un proces ideal, tot mai multe companii iși dezvoltă singuri procese automate de dezvoltare software. Aceste procese au evoluat până la punctul în care exploatează capabiltățile oamenilor dintr-o companie și caracteristicile specifice ale produselor ce se dezvolta în aceea companie.

Ciclul de viață al unui produs definește durată medie a acestuia de la analiză până la punerea acestuia în funcțiune: produsele se nasc, se dezvoltă ajung la maturitate și apoi îmbătrânesc. În funcție de perioada de viață în care se află depinde și prețul și calitatea acestuia.

Ciclul de viață al produsului software este prin urmare o schemă a apariţiei produsului software, pornind de la problema originară şi până la un produs final, care să rezolve aceea problemă originară.

Modelul acesta nu îşi arată funcționalitatea în condiţiile unor proiecte mici precum un sigur programtor ce răspunde cerințelor unui client pentru realizarea unei pagini web sau a unei aplicații foarte simple.

Însă în proiectele software din zilele noastre sunt implicate echipe mari de dezvoltatori, analişti, arhitecţi software, designeri, manageri, în multe cazuri distribuiţi în ţări sau pe continente diferite, pe perioade de timp de ordinul lunilor sau anilor, acest model teoretic începe să aibă utilitate.

El este un prim pas către separarea a ceea ce este în interiorul proiectului, din punct de vedere al tipurilor de activităţi şi din punct de vedere temporal. Este, prin urmare, un prim pas către acel „divide et impera” care permite păstrarea controlului asupra proiectului. (Referință bibleografică 1+3+4)

**Capitolul 2.**

**Logica înlănțuirii fazelor unui ciclu de viață**

În teorie există mai multe definiții care descriu detaliat toate fazele succesive ale vieții unui produs software. Una dintre aceste definiții este următoarea: "Ciclul de viață include fazele: conceptualizare, dezvoltarea ideilor proiectului, studiul de inginerie, planificarea proceselor, fabricație, operare, întreținere (reparare) și retragere".

Definiția oficială a ciclului de viață al produsului este prezentată în standardul SR EN ISO 14040: Ciclul de viață reprezintă "etape consecutive și intercorelate ale unui sistem-produs, de la achiziția materiilor prime sau generarea resurselor naturale până la post-utilizare".

Dacă ne referim la un model în domeniul marketingului, ciclul de viață este "orientat spre piață", acesta se referă la viața *comercială* a produsului adică viața produsului pe piață și descrie tendința de dezvoltare a cifrei de afaceri și a profitului unui produs, de-a lungul vieții sale *comerciale*.

In acest sens (al vieții comerciale),ciclul de viață al produsului reflectă evoluția în timp a volumului de vânzări ale produsului și reprezintă intervalul de timp dintre momentul apariției produsului pe piață și cel al dispariției sale definitive din arena schimbului de mărfuri.

Un ciclu de viață al produsului ar putea consta din următoarele etape :

1)Analiza și colectarea cerințelor

2) Proiectare

3) Implementare și dezvoltare

4) Testare și validare

5) Vânzarea și folosirea propriu zisă a produsului

6) Mentenanță

(1)

**Analiza și colectarea cerințelor**

În această etapă a proiectului are loc definirea a ceea ce trebuie dezvoltat. Obiectivul aici este să se afle nevoile clientului şi să se definească foarte clar cerinţele privind viitorul produc software. Aceasta etapă este cea mai importanta pentru project manageri și pentru persoanele care se ocupă de vânzări. Se orgaznizează cât mai multe întâlniri între managerii proiectului, agenții de vânzări și clientul pentru a stabili de comun acord aceste specificații care vor determina cum va arăta produsul final.

Aceste specificații pot fii de tipul:

Cine va folosi sistemul?

Cum vor folosi acest sistem?

Ce fel de date se vor introduce în sistem?

Ce fel de date vor ieși din sistem?

Răspunsul la aceste întrebări trebuie gasit în această fază, și reprezintă specificațile produsului. După ce se strâng toate acestea informații, sunt analizate amănunțit pentru a determina dacă sunt valide sau nu si desigur dacă pot sau nu pot fii implementate în sistemul ce urmează a fii construit.

Într-un final se crează un document cu specificațile produsului care va reprezenta modelul ce va fii urmat în etapa următoare a ciclui de viată a produsului.

**Proiectarea**

Această etapă are ca obiectiv modelarea viitorului sistem, văzut ca soluţie a problemelor determinate în faza de analiză. Dacă Analiza îşi propunea să determine ce trebuie făcut, faza de proiectare trebuie să arate cum trebuie făcut. În această fază sunt proiectate funcţionalităţile pe care viitorul sistem va trebui să le aibă.

În această etapă sistemul și arhitectura sistemului sunt pregătite urmând modelul documentului cu specificațile produsului realizat în prima etapă. Tot în această etapă se stabilesc componentele hardware și necesitățile sistemului din punct de vedere al arhitecurii sistemului. Specificațile de proiectare a sistemului vor fii folosite ca model de plecare pentru faza următoare.

**Implementare și dezvoltare**

După primirea specificaților de design a sistemului și implicit detalii despre arhitectura acestuia, munca de implementare este împărțită în mai multe etape și se dă drumul la programarea propriu zisă. Cum în această etapă are loc programarea, este o etapă ce reprezintă cea mai importantă parte pentru programator. Această etapă reprezintă de altfel și cea mai lungă etapă din ciclu de viată.

**Testarea și validarea**

După ce codul a fost implementat în totalitate, produsul este testat amănunțit în conformitate cu specificațile din prima etapă pentru asigurarea faptului ca produsul chiar rezolvă problemele pentru care a fost creat si funcționează asa cum a fost stabilit cu clientul la începutul dezvoltării. În această etapă se realizează mai multe tipuri de teste precum: testare unitară, acceptarea sistemului, test de integritate, testul final de accept.

**Deployment-ul și acceptanța**

În urma testării cu succes, produsul este trimis la client unde se instalează pentru a fii folosit. Acceptanța este faza în care clientul recepţionează sistemul software, acceptă că acesta corespunde cerinţelor lui şi îşi dă acordul pentru intrarea în faza de mentenanţă.

**Mentenanță**

După ce clientul începe folosirea propriu zisă a produsului atunci există riscul ca anumite probleme de soft sa apară în timp, probleme ce trebuiesc rezolvate tot de programatorul inițial. Acest proces în care după livrare se asigură o garanție a produsului și posibilitate de reparare ulterioară se numește Mentenață. Intrarea în faza de mentenanţă înseamnă încetarea includerii oricăror noi cerinţe în sistem şi corectarea bug-urilor (anomaliilor în funcţionare). Această fază este importantă pentru că ea constituie adesea o fază costisitoare, dar şi prea adesea ignorată. (Referință bibleografică 3+4+5)

**Capitolul 3.**

**Modeluri de cicluri de viață și alegerea unui ciclu de viață potrivit**

Modelele ciclurilor de viață reprezintă diverse procese sau metodologii ce sunt selectate pentru dezvoltarea proiectelor în fucție de ce scopul acestor acestor proiecte și de de ce și-au propus să realizeze aceste proiecte. Există foarte multe modele de cicluri de viață care au fost dezvoltate cu scopul de a îndeplini diferite cerințe și specificații. Aceste modele definesc etapele proceselor și ordinea în care acestea se execută.

Alegerea modelului ciclului de viață are un impact foarte mare în special asupra testării ce urmează a fii făcută. Va influența ce anume, unde anume și când anume vom testa produsul, influențează de asemenea și tehnica aleasă de testare.

Printre varietatea de modele de cicluri de viață definite se numără următoarele:

* Modelul cascadă
* Modelul în V
* Modelul iterativ
* Agille programming
* Modelul spirală

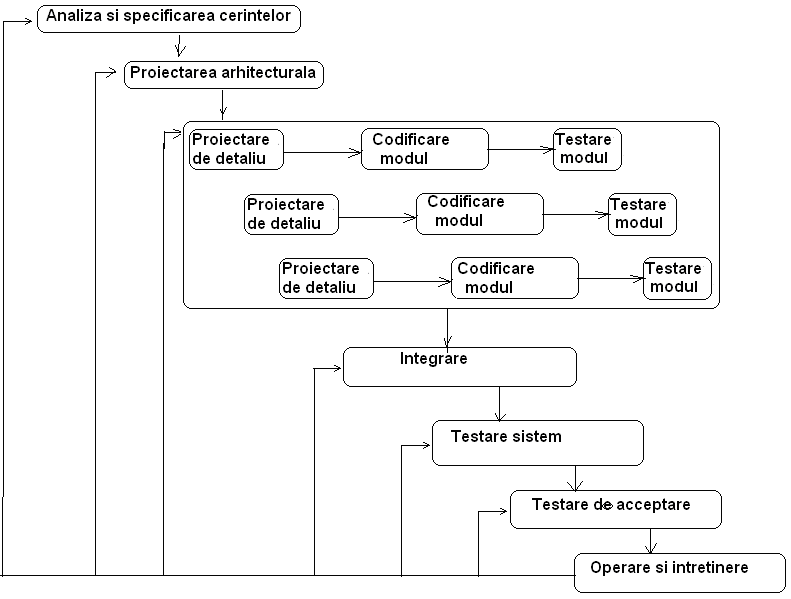
Alegerea unui model de ciclu de viață potrivit pentru un produs software este foarte importantă. Pe baza acestui model se bazează și implementarea codului și testarea acestuia ulterioară.

Companiile software aleg acest model bazându-se pe tipul de soft pe care îl implementează și pe produsul final pe care doresc să-l obțină. Cel mai vechi model folosit este reprezentat de Ciclul de viață în cascadă. În acest model testarea începe în momentul în care toată implementarea, programarea s-a încheiat. Din acest motiv există foarte multe erori și defecte care sunt determinate deabeea la sfârșit. Prin urmare costul reparației acestor erori este unul destul de ridicat. Cel mai folosit model este reprezentat de modelul agil. In modelul agil, după fiecare sprint se realizează un demo la client. În acest fel clientul poate vedea chiar din timpul implementării dacă produsul se îndreaptă către ceea ce doreau sa obțină sau nu.

(Referință bibleografică 1,3)

**Modelul ciclului de viață în cascadă**

Modelul ciclului de viață în cascadă reprezintă primul model de ciclu de viață recunoscut. Este un model simplu de înțeles și folosit. În acest model fiecare etapă trebuie finalizată complet înainte de a trece la următoarea etapă. Acest model este folosit, deobicei, pentru proiecte micuțe și care nu conține specificații incerte. La sfârșitul fiecărei etape se realizează un document în care se specifică ce s-a realizat/hotărât până în acel moment, dacă proiectul se află sau nu pe drumul cel bun și daca se dorește continuarea proiectului sau renunțarea cu totul.



Imagine: http://www.scritub.com/stiinta/informatica/Modele-ale-Procesului-de-dezvo1411914920.php

În prima sa versiune, modelul conținea numai sgeile descendente, care materializau nlnuirea etapelor; el nu prevedea și iteraiile. Sgeile ascendente au fost introduse mai trziu pentru a ilustra principiul c o etap repune n discuie numai etapa precedent.

Modelul ciclului de viață în cascadă se bazeaz pe faptul că etapele sunt foarte bine delimitate. Rezultatele produse de fiecare faz sunt documentate și evaluate cu ajutorol reviziilor care aprob trecerea de la o etapă la alta. Din nefericire, testul final a bunei sau a proastei functionri a produsului se realizează n cadrul etapei de testare, cnd este posibil testarea concret a programului final. Înaintea acestei etape au fost realizate numai documente.

În realitate s-a ajuns la concluzia c partea dificilă în acest model se datorează în principal următoarelor probleme :

- înțelegerea greșită a problemelor de către analiști și/sau client

- instabilitatea cerințelor care se modifică pe parcursul dezvoltării

-Alegeri tehnice greșite

-schimbrilor de personal.

Datorită acestor motive este nevoie de reveniri n faze anterioare ale procesului de dezvoltare. Aceste reveniri reprezintă de fapt o reflectare a realitii. Dac aceste întoarceri sunt rare i limitate la faze ocazionale, modelul n cascad poate fii folosit. Dacă nu, modelul n cascad nu corespunde realitii.

### Avantaje ale acestui model:

### Documentație foarte bună a sistemului

### Ușor de manageruit

### Dezavantaje acestui model

### Se obține foarte târziu o dovadă a unui produs funcționabil, deabeea după terminarea totală a implementării. Până în acel moment fizic există doar documente.

### Decoperirea târzie a erorilor implică un cost crescut

### Toate riscurile sunt incluse intr-un singur ciclu de dezvoltare

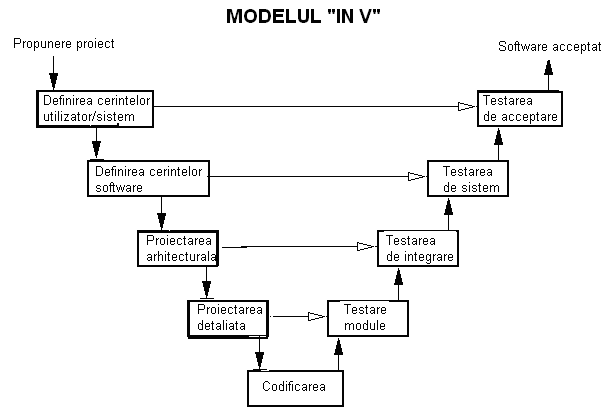
Modelul n cascad d rezultate acceptabile numai n cazul n care este efectiv posibil nlnuirea fazelor fr prea multe probleme. Se presupune că totalitatea cerinelor s fie cunoscută în totalitate i problema să fie nteleas în deplin de analiti. Trebuie de altfel ca soluia finală s fie uor de găsit de proiectani i implementarea să fie simpl - redus la generarea automat a codului plecnd de la documentele de proiectare.

### (Referință bibleografică 2,3,5,8)

### Modelul in V

Modelul ciclului de viață în V reprezintă o varianta a modelului cascada, care pune in evidenta corelarea dintre activitatile de specificare si cele de testare, inlantuirea in timp a activitatilor fiind aceeasi. V-ul vine e la validare. La fel ca și modelul în cascadă, modelul în V reprezintă un proces secvențial de execuție a proceselor de dezvoltare soft.

Fiecare fază trebuie încheiată înainte ca faza următoare să înceapă. Testarea produsului se realizează în paralel cu partea de dezvoltare in care se află produsul.



Imagine: http://www.scritub.com/stiinta/informatica/Modele-ale-Procesului-de-dezvo1411914920.php

Partea stângă a modelului reprezintă lanțul în care se produce specificarea sistemului iar partea din dreapta reprezintă lanțul de testare. Partea de jos a modelului reprezintă implementarea.

Modelul conține două tipuri de relații/asocieri între etape:

1) Linilie care formează v-ul corespund nlanuirii i eventualele iteraii din modelul cascad. Fazele se produc secvenial de la stnga la dreapta

2) Linilie orizontale indic faptul c o parte din rezultatele fazei din care provine sgeata sunt utilizate direct n etapa următoare. Cum ar fii, la finalul fazei de proiectare, cazurile de teste de integrare trebuie s fie complet descrise.

**Avantajele acestui model :**

1. Ușor și simplu de folosit.
2. Activitățile de testare precum planificarea testelor si modelelor de testare se realizează înainte de testarea propriu zisă ceea ce salvează foarte mult timp.
3. Defectele programului sunt găsite la timp și costul reparației va fii mult mai mic
4. Este util pentru proiecte micuțe unde ceințele sunt foarte bine specificate

**Dezavantajele acestui model :**

1. Foarte rigid și deloc flexibil
2. Softul este dezvoltat în timpul etapei de dezvoltare, deci înainte de această etapă nu avem prototipuri ale softului
3. Dacă se produc schimbări în timpul programării toate documentele de test și documentele cu specificații trebuiesc updatate.

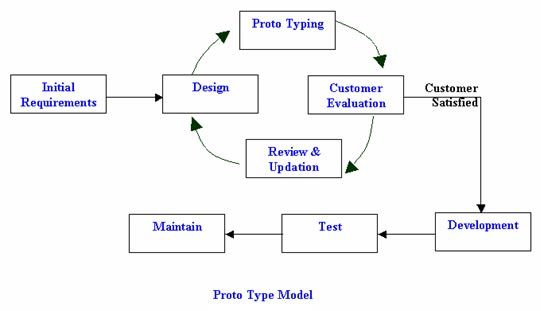
Acest model este util a fii folosit în proiecte micuțe sau medii în care cerințele sunt foarte clare și stabile de la început. De asemenea, este de preferat folosirea acestui model când avem la dispoziție resurse tehnice destul de avansate.

(Referință bibleografică 3,5,8)

### Modelul Iterativ

Modelul ciclului de viață iterativ a fost gândit în opoziție cu modelul ciclui de viață în cascadă. Ideea de bază a acestui model este: dacă un produs este construit foarte complex devine dificil de înțeles dar și de implementat și conceput într-o singură fază. Modelul desparte modelul în mai multe faze prin evoluție. Fiecare etapă urmând apoi a fii trecută prin faza de proiectare, de implementare și testare.

**Ideea de bază a acestui model** este de a îmbunătății două limitări ale modelului cascadă. În loc de a hotărî cerințele înainte de a incepe proiectarea si implementarea, este realizat un prototip pentru înțelegerea cerințelor. Acesta este dezvoltat pe baza cerințelor cunoscute în momentul actual. Dezvoltarea prototipului conține fazele de proiectare, implementare si testare, dar acestea nu sunt foarte riguroase sau formale.  
     Cu ajutorul prototipului, clientul ințelege mai bine modul in care lucrează produsul, întrucât interacționează cu acesta pe parcursul întregului ciclu de dezvoltare.  
     Acest model este preferat in cadrul sistemelor mari si complicate, pentru care este dificilă înțelegerea cerințelor de la început.



Imagine: https://learnwithkamal.wordpress.com/software-engineering/prototype-model/

O prima variantă de produs software funcțional este realizată dupa primul modul, deci avem acces la un soft funcțional foarte devreme în ciclul de viață al produsului. Fiecare modul al modelului adaugă noi funcționalități produsului anterior. Procesul se reia până obținem produsul final pe care ni l-am propus.

Ciclul de via iterativ are la bază evoluia produselor deja executabile, msurabile i deci pe evoluia elementelor concrete. El este opus modelului n cascad care se baza pe producerea de documente. Progresele se msoar prin programe demonstrabile mai degrab dect prin documente sau estimri, ca n ciclul de via n cascad;(Referință bibleografică 6)

In cursul dezvoltrii clientul poate testa produsele intermediare.

Pașii algoritmului:

* Se analizează scopul proiectului și faptul dacă este sau nu realizabil.
* Se detaliază specificațile clientului și se realizează o analiză de nivel inalt, pentru a se determina cerintele software la un nivel general.
* Se hotărăște o arhitectură generală a produsului.
* Se împart specificațile în subseturi ce pot fi implemenate în pași separați. Se hotărește planificarea temporală a acestor pași.
* Fiecare pas implementează un subset de cazuri de utilizare high level care implementează cerințele clienților El este realizat urmărind metoda în cascadă: analiza detaliata a cerintelor din subset, proiectarea, implementarea și testarea. Rezultatul este un produs care satisface un subset al cerințelor și este livrat utilizatorilor.
* Un produs tipic constă din 10-50 incremente.
* Se incepe cu o implementare simplă a unui subset al cerințelor software. Rezultă prima livrare.
* Scopul primei implementări este de a crea un produs la care utilizatorul poate reacționa. El trebuie sa pună in evidență aspectele cheie ale problemei și să furnizeze o soluție sufient de simplă pentru a fi ințeleasă și ușor de implementat.
* Fiecare nouă iterație include analiza ultimei versiuni si adăugarea de noi funcționalități, ceea ce presupune reproiectarea, codificarea si testarea. Se urmărește ca proiectarea sa fie directă, modulară, să suporte reproiectarea.
* Analiza unei iterații se bazează pe feedback-ul utilizatorului si pe instrumentele de analiză disponibile. Ea se referă la: modularitea produsului, cuplarea modulelor, utilizabilitatea, fiabilitatea, eficienta si realizarea scopurilor.
* Fiecare iterație este o variantă imbunătățită a celei anterioare, de aceea metoda se mai numește “îmbunătățirea iterativă” (iterative enhancement).
* Se utilizează măsuri pentru evaluarea evoluției sistemului. Măsurile prin care se evaluează un software sunt dificil de ineles ca valori absolute, dar schimbările valorilor lor in evoluția unui sistem sunt o bază de comparație. Astfel de măsuri sunt: numărul de defecte, efortul de actualizare, dimensiune, complexitatea, cuplarea modulelor. Schimbările diferitelor aspecte se pot monitoriza si se pot stabili limite pentru anumite măsuri, pentru a semnala probleme sau anomalii.
* Utilizarea analizei si a măsuratorilor ca ghid in procesul iterativ este o diferență majoră între aceasta metodă si metodele de “dezvoltare agilă”.Ele sunt suportul pentru determinarea eficienței procesului si a calității produsului.
* Fiecare fază a ciclului de via incremental este o reprezentare a ciclului de via n cascad la o scar mai mic. Scopul unei iteraii este hotărât pe baza analizei iteraiilor anterioare. Documentaia este realizat treptat, pe parcursul fiecrei iteraii. (Referință bibleografică 2,3,5,8)

**Avantajele acestui model:**

1. In fiecare etapa este livrat un produs executabil, care satisface o parte din cerintele utilizator. Opus modelului cascada in care se elaboreaza documente.
2. Prototipurile sunt livrate clientului/utilizatorilor.
3. Feedback-ul utilizatorilor este distribuit pe intreg parcursul dezvoltarii.
4. În cazul aparitiei unor schimbari in cerinte acestea pot fi incoporate in urmatorul prototip.

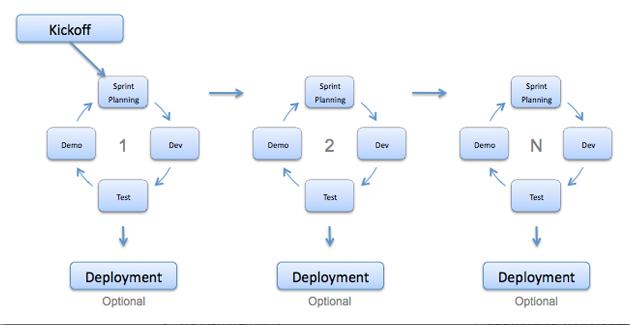
**Dezavantajele acestui model**

1. Necesită o foarte amănunțită planificare
2. Necesită o definirea clară și completă a sistemului înainte ca acesta să poată fii spart în module și incrementat pe bucăți
3. Costul total este mai mare decât la modelul în cascadă.

Acest model poate fii folosit atunci când cerințele întregului model sunt complete foarte bine definite și clar înțelese. Caracteristicile principale trebuiesc să fie foarte bine stabilite, alte detalii mai pot fii schimbate în timp. Se folosesc atunci când există o cerere de livrare a produsului într-un timp limitat. (Referință bibleografică 5,7,8)

**Agile software development**

Modelele de dezoltare software agile sunt tot un fel de model de ciclu de viață cu incrementare. Softul este dezvoltat prin metode prin cicluri rapide, iterative. Rezultatul este dat de produse micute care se realizează după fiecare build și care îmbunătățesc mereu produsul anterior. Fiecare produs lansat este testat riguros pentru a se asigura faptul că se menține calitatea softului. Este folosit mai ales pentru aplicații care au nevoie de un timp exact.



Imagine: http://istqbexamcertification.com/what-is-agile-model-advantages-disadvantages-and-when-to-use-it/

**Avantajele acestui model:**

1. Foarte satisfăcător pentru client care poate primi continuu produse soft îmbunătățite
2. Oamenii și interacțiunile sunt puși în evidență spre deosebire de procesul în sine și uneltele de lucru. Clienții, dezvoltatorii și testerii trebuie să interacționeze în permanentă.
3. Produsele soft sunt livrate în intervale de timp foarte scurt
4. Conversațile fată în față sunt cele mai bune forme de comunicare
5. Pot avea loc colaborări zilnice între dezvoltatori și clienți
6. Se pot face modificări în soft oricât de avansat e produsul fără probleme

**Dezavantajele acestui model**

1. Nu se pune suficient accent pe pe documentație și design
2. Proiectul poate să o ia pe o cale greșită dacă reprezentatul clientului nu exprimă cu exactitate ce anume dorește clientul la sfârșit
3. Numai programatorii seniori sunt capabili să ia decizii referitoare la procesul de dezvoltare. Nu este un domeniu ușor accesibil pentru programtorii juniori decât dacă lucrează cu ajutorul unui programtor senior.

Acest model este folosit când este nevoie de implementarea unor modificări noi. Oferă o libertate foarte mare schimbărilor. Aceste schimbări sunt introduse cu un preț foarte mic datorită rapidității cu care sunt livrate noi variante de produs. Pentru a implementa o nouă funcționalitate, programtorii pierd numai munca pe câteva zile sau chiar cateva ore.

Spre deosibire de modelul în cascadă, în modelul agil se pune foarte puțin accent pe planificare înainte de începerea proiectului. Acest model presupune ca dorințele și cerințele clientului vor fii într-o schimbare permanentă datorita dinamicității it-ului. Modificările pot fii mereu discutate și implementate sau anumite părți pot fii scoase în totalitate în urma discuților cu clienții. Prin urmare șansa ca la final clientul să obțină proiectul dorit este mult mai mare.

Este una dintre cauzele principale pentru care proiectele de realizare a produselor software nu reușesc respecte termenul de livrare sau nu sunt acceptate de client.

În metoda in cascadă progresul realizării unui produs era cu greu urmărit de client. Deși primește notificări detaliate despre ce se implementează, datorită faptlui că el nu vede nimic până la finalul produsului ( până la faza de operare ), atunci îi era greu să aibă încredere în progresul proiectului sau să sprijine eventuale decizii care trebuie luate pentru a finaliza proiectul.

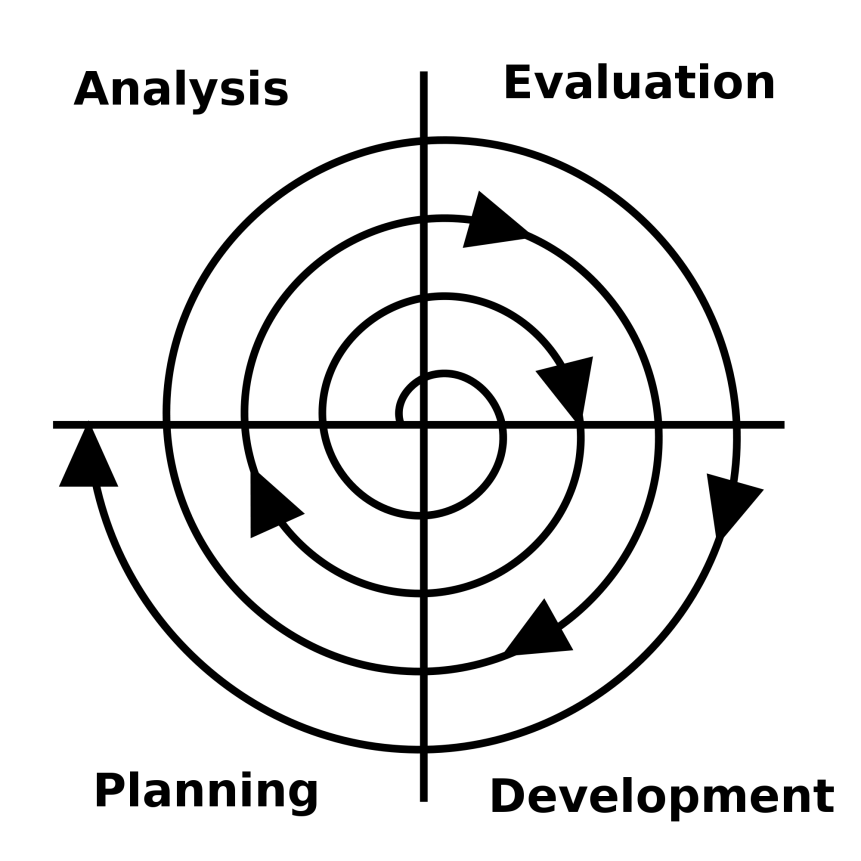
Agile a fost găndit să rezolve aceste două probleme pornind de la câteva idei de bază:  
1. Prioritatea este satisfacerea clientului prin crearea de software valoros și funcțional  
2. Schimbările sunt binevenite, indiferent de etapa de dezvoltare la care se află proiectul  
3. Software funcțional este principala măsură a progresului  
4. Simplicitatea este esențială  
Ideea principală a acestui model este că se lucrează împreună cu clientul. (3, 5, 6, 8)

**Modelul Spirală**

Modelul spirală este asemănător cu modelul iterativ, cu o pondere mai mare pusă pe analiza de risc. Modelul spirală are patru etape

1. Planificare
2. Analiză de risc
3. Dezvoltare
4. Evaluare

Un produs software trece prin aceste etape în mod repetat. Linia de bază a spiralei începe în fază de planificare, se colectează specificațile și se evaluează riscul. Fiecare sublinie se construiește plecând de la linia de bază.



Imagine: http://en.wikibooks.org/wiki/Introduction\_to\_Software\_Engineering/Process/Methodology

Etapa de planificare

În această etapă a proiectului are loc definirea a ceea ce trebuie dezvoltat. Obiectivul aici este să se afle nevoile clientului şi să se definească foarte clar cerinţele privind viitorul produc software. Se stabilesc exact cerințele în conformitate cu dorințele clientului

Etapa de analiza a riscului

În această etapă a proiectului, se realizează un proces pentru a se descoperi riscurile și a se încerca să se gasească soluții alternative. La sfârșitul acestei faze se crează un prototip. Dacă s-a descoperit vreun risc în timpul analizei de risc atunci soluțile sugerate sunt implementate

Etapa de dezvoltare

În această etapă softul este implementat în paralel cu testarea la sfârșitul fiecarei faze.

Etapa de evaluare

În această etapă clientul poate testa varianta produsului înainte ca acesta să treacă la o nouă spirală.

**Avantajele acestui model:**

1. Riscurile sunt evitate
2. Foarte bun pentru proiecte mari/critice
3. Necisită foarte bun control si documentație
4. Funcționalități noi pot fii introduse între timp
5. Soft funcțional este produs în primii pași ai ciclului

**Dezavantajele acestui model:**

1. Costul este destul de mare
2. Analiza riscului necesită personal cu experiență
3. Succesul proiectului depinde foarte mult de rezultatele analizei riscului
4. Nu funcționează bine pentru proiecte mici

Acest model este indicat a fii folost atunci când pentru un proiect, riscurile și costurile sunt foarte importante. Se folosește numai pentru proiecte mari. Se recomandă atunci când clienții nu sunt foarte siguri de ceea ce își doresc de la produsul final. Indicat atunci când cerințele sunt complexe, când produsul este nou și pentru modificări mari.

(Referință bibleografică 3,5,7,8)

**Capitolul 4.**

**Concluzii**

În concluzie putem spune ca fiecare model de ciclu de viață prezentat nu este nici mai bun, nici mai prost decât altul doar se adresează unor tipuri de produse diferite.

Modelul n cascad d rezultate acceptabile numai n cazul n care este efectiv posibil nlnuirea fazelor fr prea multe probleme. Se presupune că totalitatea cerinelor s fie cunoscută în totalitate i problema să fie nteleas în deplin de analiti. Trebuie de altfel ca soluia finală s fie uor de găsit de proiectani i implementarea să fie simpl - redus la generarea automat a codului plecnd de la documentele de proiectare.

Modelul în V este util a fii folosit în proiecte micuțe sau medii în care cerințele sunt foarte clare și stabile de la început. De asemenea, este de preferat folosirea acestui model când avem la dispoziție resurse tehnice destul de avansate.

Modelul incremental poate fii folosit atunci când cerințele întregului model sunt complete foarte bine definite și clar înțelese. Caracteristicile principale trebuiesc să fie foarte bine stabilite, alte detalii mai pot fii schimbate în timp. Se folosesc atunci când există o cerere de livrare a produsului într-un timp limitat.

Modelul de agile programming este folosit când este nevoie de implementarea unor modificări noi. Oferă o libertate foarte mare schimbărilor. Aceste schimbări sunt introduse cu un preț foarte mic datorită rapidității cu care sunt livrate noi variante de produs. Pentru a implementa o nouă funcționalitate, programtorii pierd numai munca pe câteva zile sau chiar cateva ore.

Modelul spirala este indicat a fii folost atunci când pentru un proiect, riscurile și costurile sunt foarte importante. Se folosește numai pentru proiecte mari. Se recomandă atunci când clienții nu sunt foarte siguri de ceea ce își doresc de la produsul final. Indicat atunci când cerințele sunt complexe, când produsul este nou și pentru modificări mari.

Bibleografie

1. <http://ro.wikipedia.org/wiki/Ciclu_de_via%C8%9B%C4%83_al_produsului>
2. <http://stst.elia.pub.ro/news/IS/IS_PPT/Software%20engineering%208th%20ed.pdf>
3. <http://www.computerworld.com/article/2576450/app-development/system-development-life-cycle.html>
4. <http://en.wikipedia.org/wiki/Software_development_process>
5. [http://istqbexamcertification.com](http://istqbexamcertification.com/what-is-prototype-model-advantages-disadvantages-and-when-to-use-it/)
6. <http://andrei.clubcisco.ro/cursuri/>
7. <http://en.wikibooks.org/wiki/Introduction_to_Software_Engineering/Process/Methodology>
8. <http://www.scritub.com/stiinta/informatica/Modele-ale-Procesului-de-dezvo1411914920.php>