Tema 2 – Echipa de proiect. Ciclul de viață al dezvoltării software. Modele de dezvoltare software.

Proiectul este un proces unic, constând într-un ansamblu de activități coordonate și controlate, cu date de început și de final, întreprinse pentru a atinge un obiectiv conform cerințelor specifice, inclusiv constrângerile de timp, cost și resurse.

Caracteristicile proiectului:

• Temporar - Această caracteristică cheie înseamnă că fiecare proiect are un început finit și un sfârșit finit. Începutul este momentul în care proiectul este inițiat și conceptul său este dezvoltat. Sfârșitul este atins atunci când toate obiectivele proiectului au fost îndeplinite (sau neîndeplinite dacă este evident că proiectul nu poate fi finalizat - atunci este întrerupt).

• Livrabile Unice - Orice proiect își propune să producă unele livrabile care pot fi un produs, serviciu sau alt rezultat. Livrabilele ar trebui să abordeze o problemă sau o nevoie analizată înainte de începerea proiectului.

• Elaborare progresivă - Odată cu progresul unui proiect, investigația și îmbunătățirea continuă devin disponibile, iar toate acestea permit producerea de planuri mai precise și mai cuprinzătoare. Această caracteristică cheie înseamnă că iterațiile succesive ale proceselor de planificare duc la dezvoltarea de soluții mai eficiente pentru progresul și dezvoltarea proiectelor.

Managementul proiectului este disciplina de planificare, organizare, motivare și control al resurselor pentru a atinge obiectivele specifice ale proiectului.

Provocarea principală - atingerea tuturor obiectivelor și scopurilor proiectului. Principalele constrângeri - domeniul de aplicare, timpul, calitatea și bugetul.

Echipa de proiect

Echipa Proiectului este grupul responsabil pentru planificarea și executarea proiectului.

1. Sponsorul Proiectului (Clientul)

Sponsorul Proiectului / Directorul de Proiect este un manager cu un interes demonstrabil în rezultatul proiectului, care este responsabil pentru obținerea autorității de cheltuieli și a resurselor pentru proiect. De obicei, este un manager senior sau șeful companiei/diviziei.

Roluri:

• Susținerea proiectului atât intern, cât și extern

• Promovarea proiectului

• Obținerea bugetelor pentru proiect

• Acceptarea responsabilității pentru problemele escaladate de la managerul de proiect

• Aprobarea documentelor precum documentul de inițiere a proiectului

• Susținerea Managerului de Proiect în gestionarea proiectului

2. Managerul de Proiect

Managerul de Proiect (PM) este persoana responsabilă pentru îndeplinirea obiectivelor declarate ale proiectului, asigurându-se că Echipa Proiectului finalizează proiectul.

Principalele responsabilități sunt:

• Dezvoltarea planului de proiect

• Furnizarea de rapoarte pentru părțile interesate ale proiectului

• Gestionarea echipei de proiect

• Gestionarea riscurilor proiectului

• Gestionarea programului proiectului

• Gestionarea conflictelor proiectului

3. Managerul de Produs

Un Manager de Produs încearcă să afle nevoile clienților și să dezvolte un produs pentru a le satisface. Munca sa include, de asemenea, asigurarea că produsul și eforturile de marketing susțin strategia și obiectivele generale ale companiei. Managementul Produsului definește obiectivele strategice de afaceri care inițiază proiecte discrete.

Principalele responsabilități sunt:

• Monitorizarea programului proiectului și raportarea stării acestuia.

• Inițierea deciziilor critice pentru lansarea produsului.

• Se ocupă de marketingul produsului (produsul ar trebui să corespundă cerințelor utilizatorilor, posibilităților companiei).

Diferența între Managerul de Proiect și Managerul de Produs:

• Scopul Managementului Proiectului:

Finalizarea sarcinii în cadrul bugetului și termenelor limită

• Scopul Managementului Produsului:

Crearea unui produs care va avea succes pe piață

4. Analistul de Afaceri (BA)

Analizatorii de afaceri identifică nevoile de afaceri și determină soluțiile la problemele de afaceri. Responsabil pentru colectarea și crearea documentației necesare pentru produs și traducerea acestora în specificații funcționale.

Principalele responsabilități sunt:

• Scrierea și gestionarea cerințelor și specificațiilor

• Traducerea nevoilor de afaceri către echipa de proiect

• Înțelegerea afacerii și formularea de recomandări pentru îmbunătățiri

5. Arhitecții de Sistem (SA)

Un arhitect de sistem este un designer de nivel înalt al unui sistem care urmează să fie implementat. O responsabilitate cheie a arhitectului este proiectarea arhitecturii software, adică luarea deciziilor cheie de proiectare referitoare la structura internă a unui sistem software și la interfețele sale tehnice.

Principalele responsabilități sunt:

• Stabilirea structurii de bază a unui sistem

• Definirea caracteristicilor și elementelor esențiale de proiectare

• Furnizarea viziunii inginerești asupra funcțiilor și scopului sistemului în viziunea utilizatorilor

• Deciderea formatelor de stocare și a protocoalelor de transmitere a datelor

• Dezvoltarea standardelor de codificare

6. Dezvoltare

Grupul care proiectează și codează produsul software.

Principalele responsabilități sunt:

• Analiza cerințelor

• Proiectarea software-ului

• Codificarea

• Participarea la lansările software și la activitățile post-lansare

• Mentenanța produsului

• Liderul de Dezvoltare

Șeful echipei de dezvoltare sau parte din ea, coordonator și expert. Distribuie sarcinile între membrii echipei, controlează executarea sarcinilor atribuite, rezolvă problemele tehnice și organizaționale ale echipei, etc.

• Dezvoltatorii

Dezvoltatorii decid asupra soluțiilor tehnice care pot fi implementate și utilizate.

Creează un produs care să îndeplinească specificațiile și așteptările clientului.

7. Asigurarea Calității (QA)

Echipa de QA se asigură de acuratețea și calitatea produselor conform cerințelor acestora.

Principalele sarcini includ:

• Măsurarea calității proceselor pentru a crea un produs de calitate

• Asigurarea respectării standardelor și procedurilor convenite

• Verificarea și rezolvarea problemelor identificate

• Verificarea faptului că un produs îndeplinește cerințele care au ghidat designul și dezvoltarea sa, funcționează conform așteptărilor și satisface nevoile părților interesate

• ***Manager de Testare (lider de testare):***

Expert(ți) în planificarea și controlul testării, cu cunoștințe și experiență în domeniile testării software, managementului calității, managementului proiectelor și managementului personalului. Sarcinile tipice pot include următoarele:

• Scrierea și coordonarea politicii de testare pentru organizație

• Dezvoltarea abordării și planului de testare

• Reprezentarea perspectivei de testare în proiect

• Achiziționarea resurselor de testare

• Selecția și introducerea strategiilor și metodelor de testare potrivite, introducerea sau îmbunătățirea instrumentelor de testare, organizarea instruirii în instrumente, decizia privind mediul de testare și automatizarea testelor

• Introducerea sau optimizarea proceselor de suport (de exemplu, managementul problemelor, managementul configurației) pentru a putea urmări schimbările și pentru a asigura reproductibilitatea testelor

• Introducerea, utilizarea și evaluarea metricilor definite în planul de testare

• Adaptarea regulată a planurilor de testare în funcție de rezultatele testelor și progresul testării

• Identificarea metricilor potrivite pentru măsurarea progresului testării și evaluarea calității testării și a produsului

• Scrierea și comunicarea rapoartelor de testare

• ***Tester:***

Expert(ți) în executarea testelor și raportarea eșecurilor (cunoștințe de bază despre IT, cunoștințe de bază despre testare, utilizarea instrumentelor de testare, înțelegerea obiectului de testare). Sarcinile tipice includ:

• Revizuirea planurilor și cazurilor de testare

• Crearea cazurilor de testare

• Actualizarea cazurilor de testare dacă este necesar

• Executarea cazurilor de testare

• Utilizarea instrumentelor de testare și a instrumentelor de monitorizare a testelor (de exemplu, pentru măsurarea performanței)

• Înregistrarea și retestarea defectelor

• Documentarea rezultatelor (crearea diverselor rapoarte)

• Proiectant de Testare (analist de testare):

Expert(ți) în metode de testare și specificație a testelor, cu cunoștințe și experiență în domeniile testării software, ingineriei software și metodelor (formale) de specificație. Sarcinile tipice pot include următoarele:

• Revizuirea cerințelor, specificațiilor și modelelor pentru testabilitate și pentru a proiecta cazuri de testare

• Crearea specificațiilor de testare

• Pregătirea și achiziționarea datelor de testare

• Inginer de Automatizare a Testelor:

Expert(ți) în automatizarea testelor, cu cunoștințe despre testarea de bază, experiență în programare și cunoștințe profunde despre instrumentele de testare și limbajele de scriptare. Automatizează teste după cum este necesar, folosind instrumentele de testare disponibile pentru proiect.

8. Ops

Ops (Operațiuni IT) sunt responsabile pentru suportul tehnic al proiectului: configurarea serverelor, pregătirea mediilor de testare.

9. Manager de Lansare

Managementul lansării este domeniul care se concentrează pe coordonarea părților din diversele livrabile ale produsului care trebuie să se unească pentru a funcționa ca un pachet de lansare integrat. Managementul lansării ajută la coordonarea cronologiilor produselor, asigurându-se că acestea sunt planificate și gestionate astfel încât dependențele viitoare să poată fi reunite într-un mod oportun.

Managerul de Lansare trebuie să interacționeze și să comunice zilnic cu Managerii de Testare, Managerii de Dezvoltare, Ops IT și, desigur, cu PMO. De asemenea, un Manager de Lansare trebuie să aibă încredere suficientă pentru a comunica superiorilor și pentru a furniza rapoarte, precum și pentru a actualiza periodic Managementul IT superior, precum și managementul de afaceri.

Fiecare proiect de dezvoltare software ar trebui să fie planificat și executat folosind un model de ciclu de viață ales în prealabil.

Lipsa unei organizari bine structurate va avea ca rezultat costuri suplimentare, necesare realizarii unui proiect de dezvoltare software, in timp ce compania de dezvoltare software nu va respecta termenele si bugetul stabilit de clientii sai.

Ciclul de viață al dezvoltării software, pe scurt SDLC, este o secvență bine definită și structurată de etape în ingineria software pentru a dezvolta produsul software dorit.

Etapele ciclului de viață al dezvoltării software:

1 – Inițierea: Acesta este primul pas în care utilizatorul inițiază cererea pentru un produs software dorit. El contactează furnizorul de servicii și încearcă să negocieze condițiile.

2 – Analiză, planificare: Etapa de analiză a procesului SDLC implică definirea obiectivelor proiectului ca funcții.

3 – Cerințe și specificații: Cerințele tehnice sunt adunate în această fază. Această fază este punctul central al managerilor de proiect și al părților interesate. Sunt organizate întâlniri cu manageri, părțile interesate și utilizatori pentru a stabili cerințele precum; Cine va folosi sistemul? Cum vor folosi sistemul? Ce date ar trebui introduse în sistem? Ce date ar trebui să fie scoase de sistem?

4 – Proiectare: În această fază, proiectarea sistemului și a software-ului este pregătită din specificațiile cerințelor. Designul sistemului ajută la specificarea cerințelor hardware și de sistem și, de asemenea, ajută la definirea arhitecturii generale a sistemului.

5 – Dezvoltare/Implementare: La primirea documentelor de proiectare a sistemului, munca este împărțită în module/unități și începe codificarea efectivă. Deoarece, în această fază, codul este produs,

astfel încât acesta este principalul obiectiv pentru dezvoltator. Aceasta este cea mai lungă fază a ciclului de viață al dezvoltării software.

6 - Testare: După ce codul este dezvoltat, acesta este testat în raport cu cerințele pentru a se asigura că produsul îndeplinește de fapt nevoile abordate inițial. În această fază sunt efectuate toate tipurile de testare.

7 - Livrare/Implementare: După testarea cu succes, produsul este livrat / implementat clientului pentru testarea de acceptare a utilizatorului.

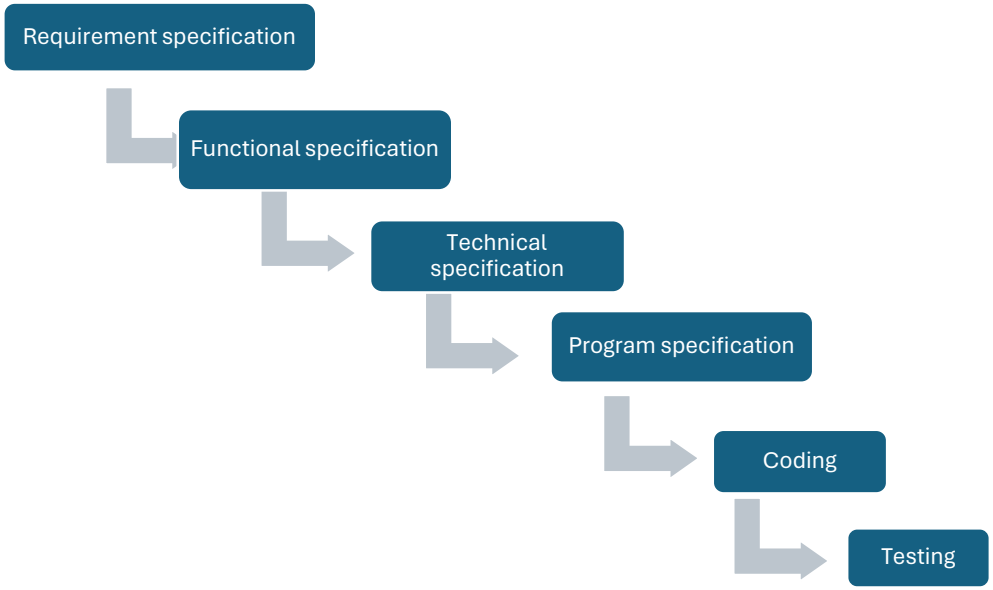
De îndată ce produsul este oferit clienților, aceștia vor face mai întâi testarea beta. Dacă sunt necesare modificări sau dacă sunt detectate erori, atunci o vor raporta echipei de testare. Odată ce aceste modificări sunt făcute sau erorile sunt remediate, va avea loc implementarea finală (de producție).

8 – Asistență/Mentenata: Odată ce clienții încep să folosească sistemul dezvoltat, problemele reale apar și trebuie rezolvate din când în când. Acest proces de triere și rezolvare a problemelor și interogărilor Post Go Live este cunoscut sub numele de întreținere.

Există diferite abordări de dezvoltare software definite și proiectate care sunt utilizate/utilizate în timpul procesului de dezvoltare a software-ului, aceste abordări sunt denumite și „Modele de proces de dezvoltare software” (de exemplu, model cascadă, model incremental, model V, model iterativ, model RAD, Model agil, model spiralat, model prototip etc.). Fiecare model de proces urmează un anumit ciclu de viață pentru a asigura succesul în procesul de dezvoltare a software-ului.

Waterfall model

Modelul Cascada a fost primul model de proces care a fost introdus. Este denumit și model de ciclu de **viață liniar-secvențial**. Este foarte simplu de înțeles și utilizat. Într-un model în cascadă, fiecare fază trebuie finalizată complet înainte ca următoarea fază să poată începe. Acest tip de model este folosit practic pentru proiectul care este mic și nu există cerințe incerte. La sfârșitul fiecărei etape, are loc o revizuire pentru a determina dacă proiectul este pe drumul cel bun și dacă să continue sau să renunțe la proiect. În acest model, testarea începe numai după finalizarea dezvoltării. În modelul în cascadă, fazele nu se suprapun.



Advantages of waterfall model:

- Acest model este simplu și ușor de înțeles și utilizat.
- Este ușor de gestionat datorită rigidității modelului – fiecare fază are livrabile specifice și un proces de revizuire.
- În acest model, fazele sunt procesate și finalizate una câte una. Fazele nu se suprapun.
- Modelul cascadă funcționează bine pentru proiecte mai mici unde cerințele sunt foarte bine înțelese.

Disadvantages of waterfall model:

• Odată ce o aplicație este în faza de testare, este foarte dificil să te întorci și să schimbi ceva care nu a fost bine gândit în faza de concept.

• Nu este produs niciun software funcțional până târziu în timpul ciclului de viață.

• Cantități mari de risc și incertitudine.

• Nu este un model bun pentru proiecte complexe și orientate pe obiecte.

• Model slab pentru proiecte lungi și aflate în derulare.

• Nu este potrivit pentru proiectele în care cerințele prezintă un risc moderat spre mare de modificare.

When to use the waterfall model:

Acest model este utilizat numai atunci când cerințele sunt foarte bine cunoscute, clare și fixe.

• Definiția produsului este stabilă.

• Tehnologia este înțeleasă.

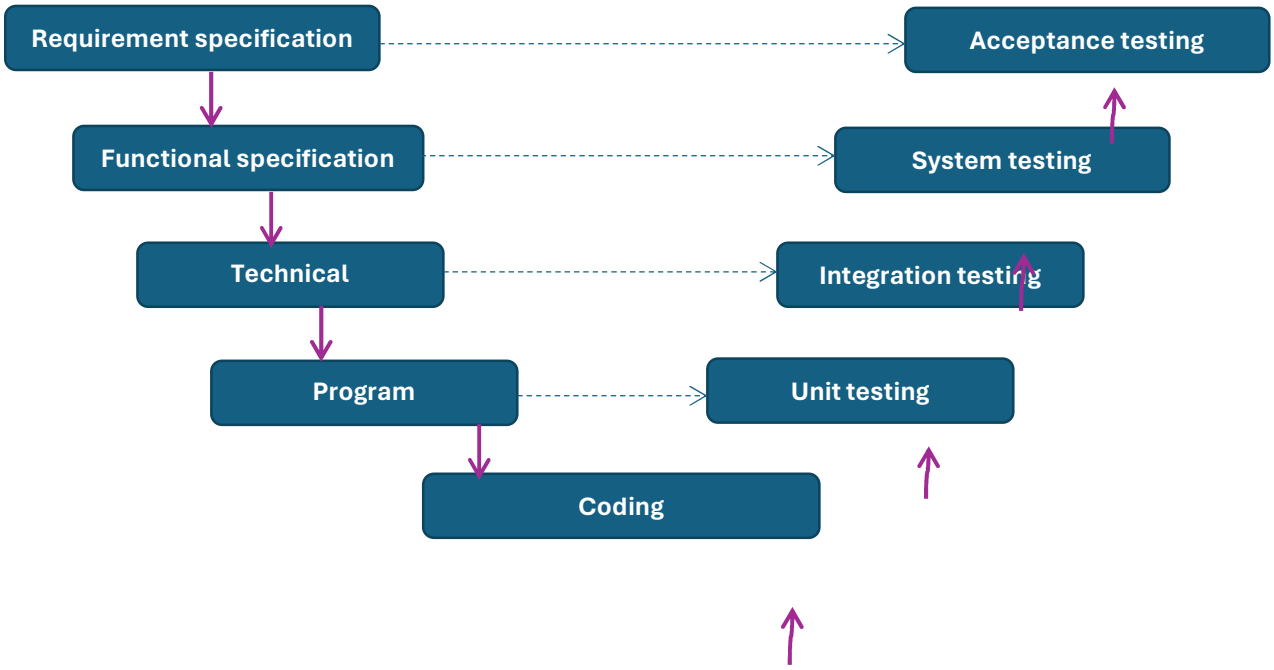
• Resursele ample cu expertiza necesară sunt disponibile gratuit

• Proiectul este scurt.

Foarte rar, acțiunea de intrare a clientului este implicată în timpul dezvoltării produsului. Odată ce produsul este gata, atunci numai acesta poate fi demonstrat pentru utilizatorii finali. Odată ce produsul este dezvoltat și dacă apare vreo defecțiune, costul remedierii unor astfel de probleme este foarte mare, deoarece trebuie să actualizăm peste tot, de la document până la logică.

V – model

Ideea principală din spatele modelului V general este că sarcinile de dezvoltare și testare sunt activități corespunzătoare de importanță egală. Cele două ramuri ale lui V simbolizează acest lucru.



Ramura stângă reprezintă procesul de dezvoltare. În timpul dezvoltării, sistemul este treptat proiectat și, în final, programat.

Ramura dreaptă reprezintă procesul de integrare și testare; elementele programului sunt asamblate succesiv pentru a forma subsisteme mai mari (integrare), iar funcționalitatea lor este testată.

Modelul V demonstrează relațiile dintre fiecare fază a ciclului de viață de dezvoltare și faza asociată de testare.

Specificația cerințelor - Nevoile și cerințele clientului sau viitorului utilizator al sistemului sunt adunate, specificate și aprobate. Astfel, se definește scopul sistemului și caracteristicile dorite.

Specificație funcțională - Acest pas mapează cerințele pe funcțiile și dialogurile noului sistem.

Specificație tehnică - Acest pas proiectează implementarea sistemului. Aceasta include definirea interfețelor cu mediul de sistem și descompunerea sistemului în subsisteme mai mici, ușor de înțeles (arhitectura sistemului). Fiecare subsistem poate fi apoi dezvoltat cât mai independent posibil.

Specificația programului - Acest pas definește fiecare subsistem, inclusiv sarcina, comportamentul, structura interioară și interfețele cu alte subsisteme.

Codare - Fiecare componentă specificată (modul, unitate, clasă) este codificată într-un limbaj de programare.

Astfel, pentru fiecare specificație și nivel de construcție, ramura dreaptă a modelului V definește un nivel de testare corespunzător:

Testare unitară - verifică dacă fiecare componentă (unitate) software își îndeplinește corect specificațiile.

Testare de integrare - verifică dacă grupurile de componente interacționează în modul specificat de proiectarea sistemului tehnic.

Testarea sistemului - verifică dacă sistemul în ansamblu îndeplinește cerințele specificate.

Testare de acceptare - verifică dacă sistemul îndeplinește cerințele clientului, așa cum sunt specificate în contract și/sau dacă sistemul satisface nevoile și așteptările utilizatorilor.

Advantages of V-model:

• Simplu și ușor de utilizat.

• Activitățile de testare precum planificarea, proiectarea testelor au loc cu mult înainte de codificare. Acest lucru economisește mult timp. Prin urmare, șanse mai mari de succes față de modelul în cascadă.

• Urmărirea proactivă a defectelor – adică defectele sunt găsite în stadiu incipient.

• Evită curgerea descendentă a defectelor.

• Funcționează bine pentru proiecte mici unde cerințele sunt ușor de înțeles.

Disadvantages of V-model:

• Foarte rigid și mai puțin flexibil.

• Software-ul este dezvoltat în faza de implementare, astfel încât nu sunt produse prototipuri timpurii ale software-ului.

• Dacă au loc modificări la jumătatea drumului, atunci documentele de testare împreună cu documentele cerinţelor trebuie să fie actualizate.

Aplicația modelului V este aproape aceeași cu modelul în cascadă, deoarece ambele modele sunt de tip secvenţial. Cerințele trebuie să fie foarte clare înainte de începerea proiectului, deoarece de obicei este costisitor să te întorci și să faci modificări. Acest model este utilizat în domeniul dezvoltării medicale, întrucât este un domeniu strict disciplinat.

Following are the suitable scenarios to use V-Model:

• Cerințele sunt bine definite, clar documentate și fixate.

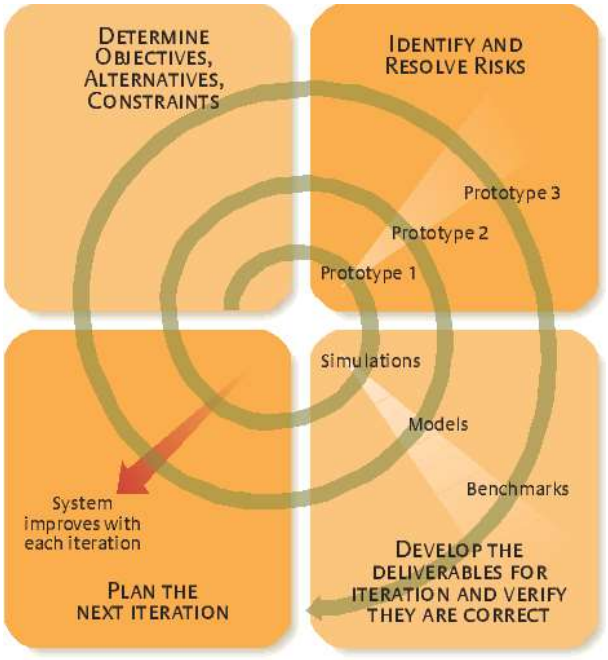
• Definiția produsului este stabilă.

• Tehnologia nu este dinamică și este bine înțeleasă de echipa de proiect.

• Nu există cerințe ambigue sau nedefinite.

• Proiectul este scurt.

Spiral Model



Adapted from B. Boehm, "A Spiral Model of Software Development and Enhancement," ACM SIGSOFT Engineering Notes 11, no. 4 (1986): 25.

Modelul spirală are patru faze: Planificare, Analiza riscurilor, Inginerie și Evaluare. Un proiect software trece în mod repetat prin aceste faze în iterații (numite Spirale în acest model). Spirala de referință, începând din faza de planificare, cerințele sunt adunate și riscul este evaluat. Fiecare spirală ulterioară se construiește pe spirala liniei de bază.

• **Planificarea cerințelor** - Cerințele sunt adunate în timpul fazei de planificare. Cerințe precum „BRS”, care este „Specificațiile cerințelor comerciale” și „SRS”, adică „Specificațiile cerințelor de sistem”.

• **Analiza și proiectarea riscurilor** - este întreprins un proces pentru a identifica riscul și soluțiile alternative. Un prototip este produs la sfârșitul fazei de analiză a riscurilor. Dacă se găsește vreun risc în timpul analizei riscului, atunci sunt sugerate și implementate soluții alternative.

• **Inginerie/Codare și testare** - În această fază este dezvoltat software-ul, împreună cu testarea la sfârșitul fazei. Prin urmare, în această fază se realizează dezvoltarea și testarea.

• **Evaluare** - Această fază permite clientului să evalueze rezultatul proiectului până în prezent înainte ca proiectul să continue la următoarea spirală.

Advantages of Spiral model:

• Cantitatea mare de analiză a riscului, prin urmare, evitarea riscului este îmbunătățită.

• Bun pentru proiecte mari și critice pentru misiune.

• Control puternic de aprobare și documentație.

• Funcționalități suplimentare pot fi adăugate la o dată ulterioară.

• Software-ul este produs la începutul ciclului de viață al software-ului.

Disadvantages of Spiral model:

• Poate fi un model costisitor de utilizat.

• Analiza riscurilor necesită o expertiză foarte specifică.

• Succesul proiectului depinde în mare măsură de faza de analiză a riscului.

• Nu funcționează bine pentru proiecte mai mici.

When to use Spiral model:

• Când este importantă evaluarea costurilor și a riscurilor

• Pentru proiecte cu risc mediu spre mare

• Angajamentul de proiect pe termen lung este neînțelept din cauza potențialelor modificări ale priorităților economice

• Utilizatorii nu sunt siguri de nevoile lor

• Cerințele sunt complexe

• Linie nouă de produse

• Sunt așteptate schimbări semnificative (cercetare și explorare)

Agile development

Modelul de dezvoltare agilă este, de asemenea, un tip de model incremental. Software-ul este dezvoltat în cicluri incrementale, rapide. Acest lucru are ca rezultat lansări incrementale mici, fiecare lansare bazându-se pe funcționalitatea anterioară. Fiecare versiune este testată temeinic pentru a se asigura că calitatea software-ului este menținută. Este folosit pentru aplicații critice de timp. Extreme Programming (XP) este în prezent unul dintre cele mai cunoscute modele de ciclu de viață de dezvoltare agilă.

Agile: Characteristics

- Munca în echipă, colaborarea și adaptabilitatea proceselor pe tot parcursul ciclului de viață al proiectului.
- Metodele agile împart sarcinile în incremente mici, cu o planificare minimă.
- Indiferent ce discipline de dezvoltare sunt necesare, fiecare echipă agilă va conține un reprezentant al clienților. La sfârșitul fiecărei iterații, părțile interesate și reprezentantul clienților analizează progresul și reevaluează prioritățile în vederea optimizării rentabilității investiției (ROI).
- Întâlniri zilnice de statut sau „stăpânire" – membrii echipei își raportează unul altuia ceea ce au făcut în ziua precedentă, ce intenționează să facă astăzi și care sunt obstacolele lor.
- Radiatorul de informații (cunoscut și sub numele de Big Visible Chart (BVC), este o reprezentare grafică mare a informațiilor despre proiect păstrată clar la vedere în spațiul de lucru partajat al unei echipe de dezvoltare agilă) prezintă un rezumat actualizat al stării unui proiect software. Un indicator luminos de construcție poate fi utilizat pentru a informa o echipă despre starea actuală a proiectului lor.

Metode de dezvoltare software agile: Crystal Clear, Dynamic Systems Development Method (DSDM), Programare extremă (XP), Scrum, dezvoltare software Lean și multe altele.

Advantages of Agile model:

• Satisfacția clienților prin livrarea rapidă și continuă a software-ului util.

• Oamenii și interacțiunile sunt accentuate mai degrabă decât procesele și instrumentele. Clienții, dezvoltatorii și testerii interacționează în mod constant între ei.

• Software-ul funcțional este livrat frecvent (în mai degrabă săptămâni decât luni).

• Conversația față în față este cea mai bună formă de comunicare.

• Cooperare strânsă, zilnică, între oamenii de afaceri și dezvoltatori.
• Atenție continuă acordată excelenței tehnice și designului bun.
• Adaptare regulată la circumstanțe în schimbare.
• Chiar și modificările tardive ale cerințelor sunt binevenite

Disadvantages of Agile model:

• În cazul unor livrabile software, în special a celor mari, este dificil de evaluat efortul necesar la începutul ciclului de viață al dezvoltării software.
• Nu se pune accent pe proiectarea și documentarea necesară.
• Proiectul poate fi îndepărtat cu ușurință dacă reprezentantul clientului nu este clar care este rezultatul final pe care îl dorește.
• Doar programatorii seniori sunt capabili să ia tipul de decizii necesare în timpul procesului de dezvoltare. Prin urmare, nu are loc pentru programatorii începători, decât dacă este combinat cu resurse experimentate.

When to use Agile model:

• Când sunt necesare noi modificări. Libertatea pe care agilul o oferă schimbării este foarte importantă. Noile modificări pot fi implementate cu un cost foarte mic datorită frecvenței de noi creșteri care sunt produse.
• Pentru a implementa o nouă caracteristică, dezvoltatorii trebuie să piardă doar munca de câteva zile, sau chiar doar ore, pentru a o derula înapoi și a o implementa.
• Spre deosebire de modelul cascadă în modelul agil, este necesară o planificare foarte limitată pentru a începe proiectul. Agile presupune că nevoile utilizatorilor finali sunt în continuă schimbare într-o lume dinamică de afaceri și IT. Modificările pot fi discutate și funcțiile pot fi efectuate recent sau eliminate pe baza feedback-ului. Acest lucru oferă în mod eficient clientului sistemul finit pe care îl dorește sau de care are nevoie.
• Atât dezvoltatorii de sisteme, cât și părțile interesate deopotrivă, constată că beneficiază, de asemenea, de mai multă libertate de timp și de opțiuni decât dacă software-ul ar fi dezvoltat într-un mod secvențial mai rigid. A avea opțiuni le oferă posibilitatea de a lăsa decizii importante până când sunt disponibile date mai multe sau mai bune sau chiar programe întregi de găzduire; ceea ce înseamnă că proiectul poate continua să avanseze fără teama de a ajunge la un impas brusc.

eXtreme Programming

Este o metodologie de dezvoltare software care are scopul de a îmbunătăți calitatea software-ului și capacitatea de răspuns la cerințele în schimbare ale clienților. Metodologia își ia numele de la ideea că elementele benefice ale practicilor tradiționale de inginerie software sunt duse la niveluri „extreme".

Există patru activități de bază pe care XP le propune pentru procesul de dezvoltare software:

1. Codificare

În XP, codificarea este considerată singurul produs important al procesului de dezvoltare a sistemului. Programatorii XP încep să genereze coduri chiar de la început.

2. Testare

Verificați dacă o funcție funcționează testând-o. XP folosește teste unitare, care sunt teste automate, iar programatorul scrie cât mai multe dintre ele pentru a încerca să spargă codul la care lucrează.

3. Ascultarea

Înțelegeți ce doresc clienții și dezvoltați soluții care se potrivesc cu nevoile și dorințele clienților cât mai aproape posibil.

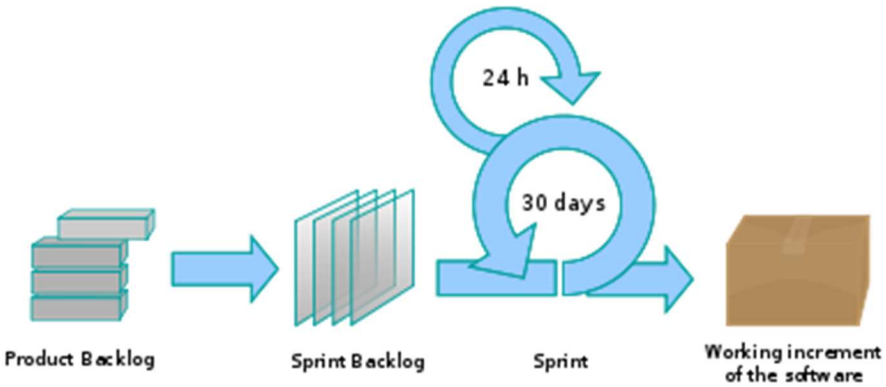
4. Proiectare

Principiul simplității XP nu înseamnă că poate exclude procesul de proiectare. Apoi, este important să se creeze o structură de proiectare care să organizeze logica în sistem, astfel încât să poată fi evitate prea multe dependențe în sistem.

Scrum methodology

Scrum este un cadru de dezvoltare software agil iterativ și incremental pentru gestionarea proiectelor software și a dezvoltării de produse sau aplicații.

Un sprint este unitatea de bază a dezvoltării în Scrum. Sprintul este un efort „timeboxed", adică este limitat la o anumită durată. Durata este fixată în avans pentru fiecare sprint și este în mod normal între 1 săptămână și 1 lună.



1) Un proprietar de produs creează o listă de dorințe cu prioritate numită backlog de produse.
2) În timpul planificării sprintului, echipa trage o mică bucată din partea de sus a listei de dorințe, un stoc de sprint și decide cum să implementeze acele piese.
3) Echipa are o anumită perioadă de timp - un sprint (de obicei, două până la patru săptămâni) - pentru a-și finaliza munca, dar se întâlnește în fiecare zi pentru a-și evalua progresul (Scrum zilnic).
4) Pe parcurs, Scrum Master menține echipa concentrată asupra obiectivului său.
5) La sfârșitul sprintului, lucrarea ar trebui să fie potențial livrabilă: gata de înmânat unui client, pusă pe raftul unui magazin sau prezentată unei părți interesate.
6) Sprintul se încheie cu o revizuire a sprintului și o retrospectivă.
7) Pe măsură ce începe următorul sprint, echipa Scrum alege o altă bucată din stocul de produse și începe să lucreze din nou