Ministerul Educației și Cercetării al Republicii Moldova
IP Colegiul “Iulia Hasdeu” din Cahul

Lecția 2 – Planificarea aplicațiilor Web

Ciclul de viață al dezvoltării software

Modele de dezvoltare software

Cuprins

Recapitulare

learningapps.org

Ce este un proiect software ?

Proiectul software este un proces unic constând dintr-un activ de activități coordonate și controlate cu date de început și de sfârșit, întreprinse pentru a atinge un obiectiv conform cerințelor specifice, inclusiv constrângeri 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.

Livrabil unic - Orice proiect își propune să livreze un rezultat unic care poate fi un produs, serviciu sau un alt rezultat. Produsele livrabile ar trebui să abordeze o problemă sau o nevoie analizată înainte de începerea proiectului.

Elaborare progresivă - Această caracteristică cheie înseamnă că iterațiile succesive ale proceselor de planificare au ca rezultat dezvoltarea de soluții mai eficiente pentru progresul și dezvoltarea proiectelor.

Echipa de proiect

Echipa de proiect este grupul responsabil de **planificarea și executarea proiectului** conform constrângerilor – domeniul de aplicare, timp, calitate și buget.

Componența și responsabilitățile echipei de proiect

Sponsorul proiectului (clientul)

Project manager
Product manager

Business analist

Arhitect de sistem

Dezvoltatori

Echipa QA (quality assurance)

Sponsorul de proiect/Directorul de proiect este un manager cu interes demonstrabil pentru rezultatul proiectului, care este responsabil pentru asigurarea autorității de cheltuieli și a resurselor pentru proiect.

Roluri:
Susținerea proiectului atât pe plan intern cât și extern
Susținerea proiectului
Obținerea bugetelor pentru proiect
Semnarea documentelor (documentul de inițiere a proiectului)
Sprijinirea managerului de proiect în gestionarea proiectului

Componența și responsabilitățile echipei de proiect

- Sponsorul proiectului (clientul)
- **Project manager**
  Product manager
- Business analist
- Arhitect de sistem
- Dezvoltatori
- Echipa QA (quality assurance)

Managerul de proiect (PM) este persoana responsabilă cu realizarea obiectivelor declarate ale proiectului, asigurându-se că echipa de proiect finalizează proiectul.

Responsabilitatile principale sunt:

Elaborarea planului de proiect
Furnizarea de rapoarte către părțile interesate din proiect
Conducerea echipei de proiect
Gestionarea riscurilor, termenelor și conflictelor proiectului

Un manager de produs încearcă să afle nevoile clienților și să dezvolte un produs care să le satisfacă. Sarcina lui include, de asemenea, să se asigure că produsele și eforturile de marketing susțin strategia și obiectivele generale ale companiei.

Responsabilitatile principale sunt:

Se ocupă de marketingul produsului (produsul trebuie să corespundă cerințelor utilizatorului, posibilităților companiei).

Componența și responsabilitățile echipei de proiect

Sponsorul proiectului (clientul)

Project manager
Product manager

Business analist

Arhitect de sistem

Dezvoltatori

Echipa QA (quality assurance)

Business analist identifică nevoile de afaceri și determină soluții la problemele de afaceri. Responsabil cu colectarea și crearea cerințelor de documentație pentru produs și transpunerea acestora în specificații funcționale.

Responsabilitatile principale sunt:
Redactarea și gestionarea cerințelor și specificațiilor
Traducerea nevoilor de afaceri către echipa de proiect
Înțelegerea afacerii și formularea de recomandări pentru îmbunătățiri

Componența și responsabilitățile echipei de proiect

Sponsorul proiectului (clientul)

Project manager
Product manager

Business analist

Arhitect de sistem

Dezvoltatori

Echipa QA (quality assurance)

Un arhitect de sistem este un proiectant de nivel înalt al unui sistem care urmează să fie implementat. O responsabilitate cheie a arhitectului este proiectarea arhitecturii software, adică a lua decizii cheie de proiectare privind structura internă a unui sistem software și interfețele sale tehnice.

Responsabilitatile principale sunt:
Stabilirea structurii de bază a unui sistem
Dezvoltarea standardelor de codare
Definește caracteristicile și elementele esențiale de proiectare
Decide asupra formatelor de stocare și a protocoalelor de transmitere a datelor

Componența și responsabilitățile echipei de proiect

Sponsorul proiectului (clientul)

Project manager
Product manager

Business analist

Arhitect de sistem

Dezvoltatori

Echipa QA (quality assurance)

Grupul care proiectează și codifică produsul software.

Responsabilitatile principale sunt:
Analiza cerinţelor
Proiectare software
Codificare
Întreținerea produsului

Dev conducător
Șeful echipei de dezvoltare sau o parte a acesteia, coordonator și expert. Distribuie sarcinile între membrii echipei, controlează executarea sarcinilor atribuite, rezolvă problemele tehnice și organizatorice ale echipei etc.

Dezvoltatori
Dezvoltatorii decid soluțiile tehnice care pot fi implementate și utilizate. Crează un produs care să corespundă specificațiilor și așteptărilor clientului.

Componența și responsabilitățile echipei de proiect

Sponsorul proiectului (clientul)

Project manager
Product manager

Business analist

Arhitect de sistem

Dezvoltatori

Echipa QA (quality assurance)

Echipa QA asigură acuratețea și calitatea produselor conform cerințelor.

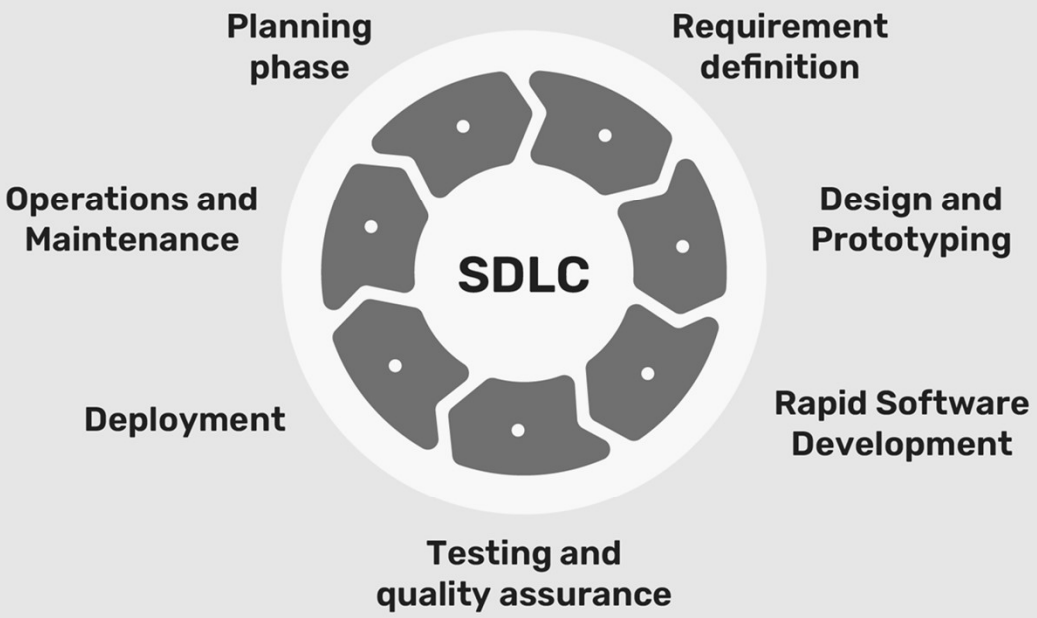
Sarcini principale:
să măsoare calitatea proceselor de creare a unui produs calitativ
asigurați-vă că sunt respectate toate standardele și procedurile convenite
asigurați-vă că problemele sunt găsite și rezolvate
să verifice dacă un produs îndeplinește cerințele care i-au ghidat proiectarea și dezvoltarea, funcționează conform așteptărilor și satisface nevoile părților interesate

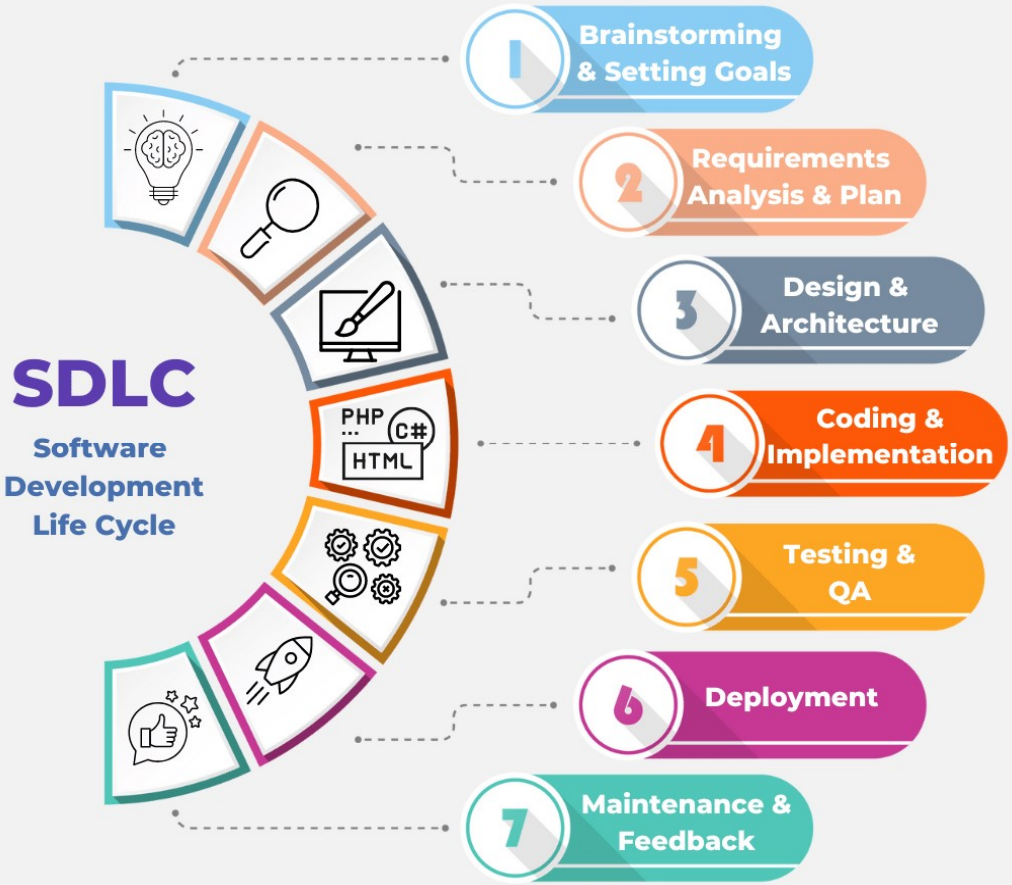
Etapele dezvoltării software

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

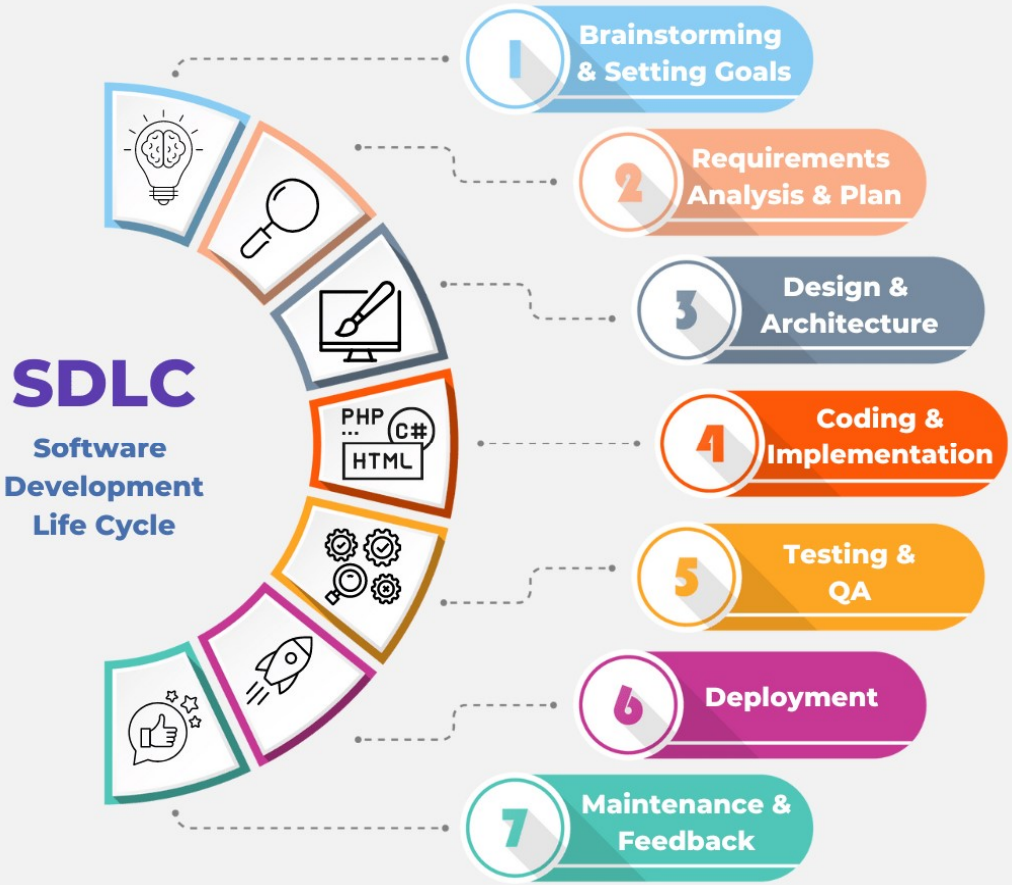




1 – Inițierea: Acesta este primul pas în care utilizatorul inițiază cererea pentru un produs software dorit. El contactează furnizorul de servicii și încearcă să negocieze condițiile.

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

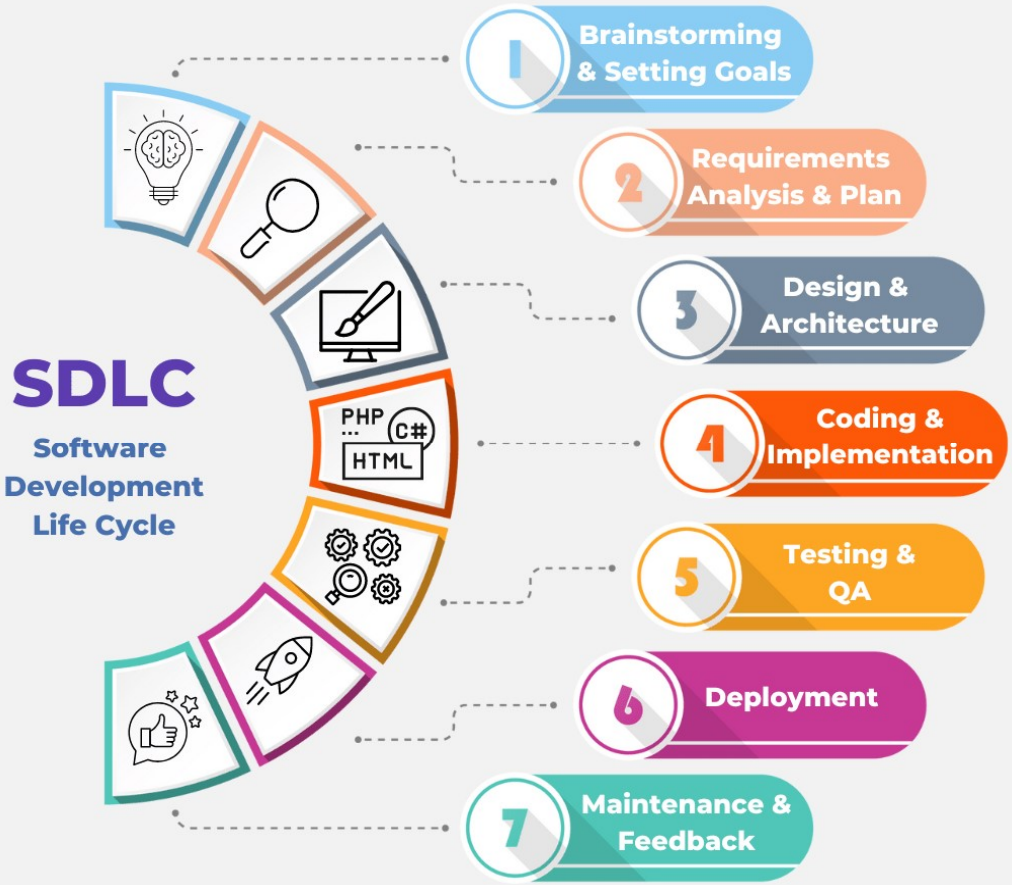
3 – Cerințe și specificații: Cerințele tehnice sunt adunate în această fază. Această fază este punctul central al managerilor de proiect și al părților interesate. Sunt organizate întâlniri cu manageri, părțile interesate și utilizatori pentru a stabili cerințele precum; Cine va folosi sistemul? Cum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

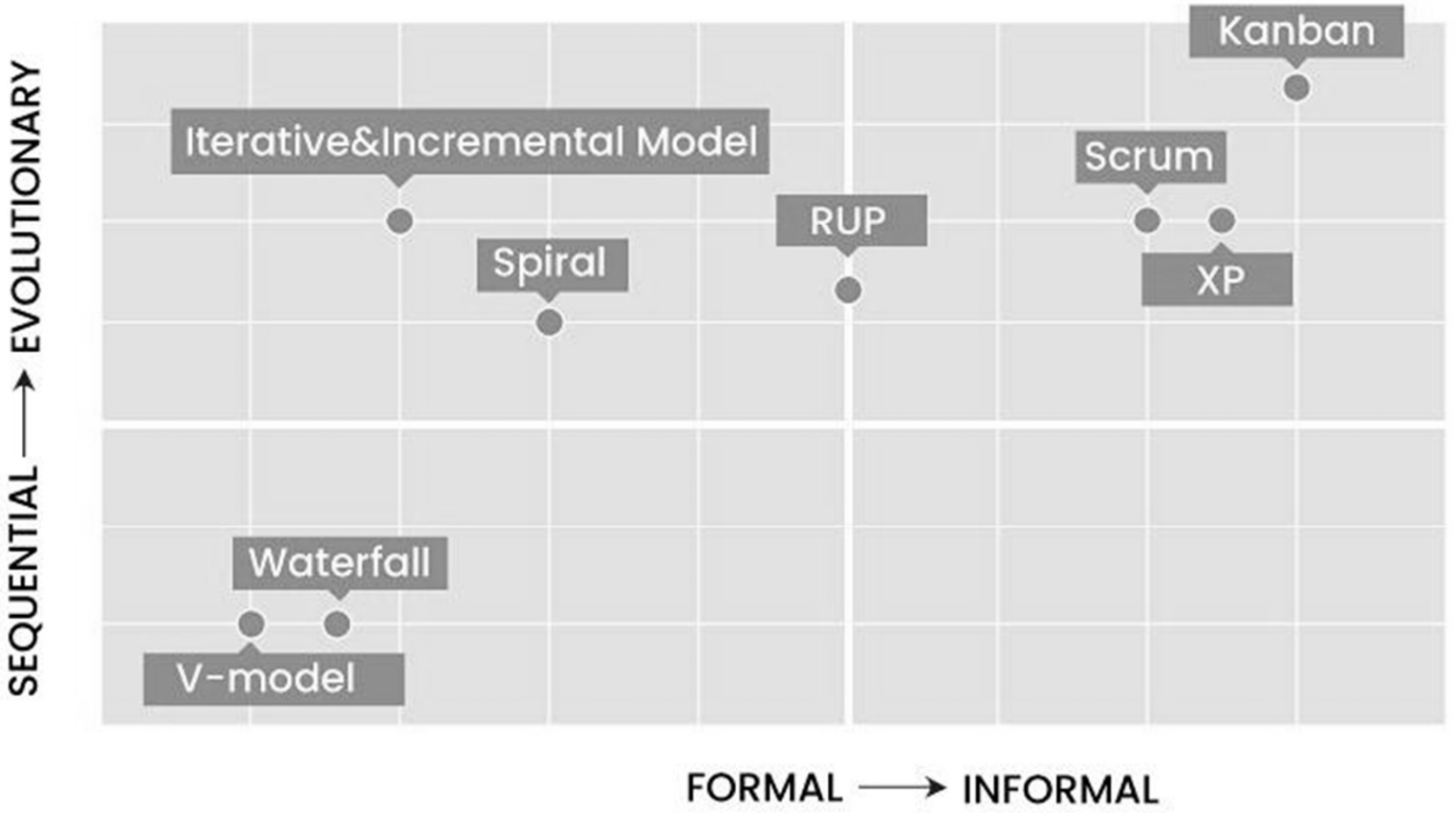


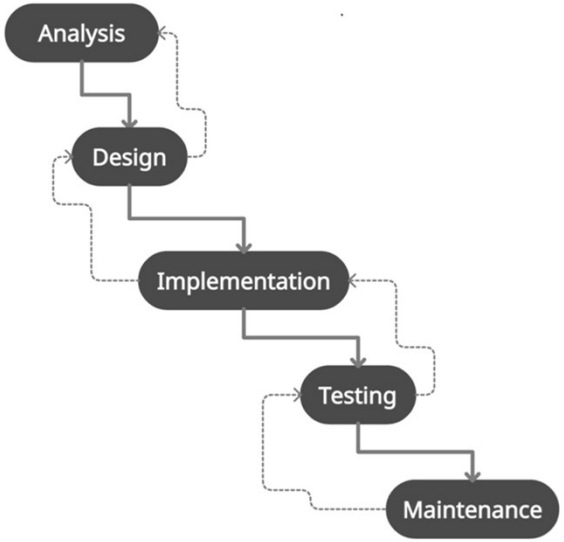
7 - Livrare/Implementare: După testarea cu succes, produsul este livrat / implementat clientului pentru testarea de acceptare a utilizatorului.
De îndată ce produsul este oferit clienților, aceștia vor face mai întâi testarea beta. Dacă sunt necesare modificări sau dacă sunt detectate erori, atunci o 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.

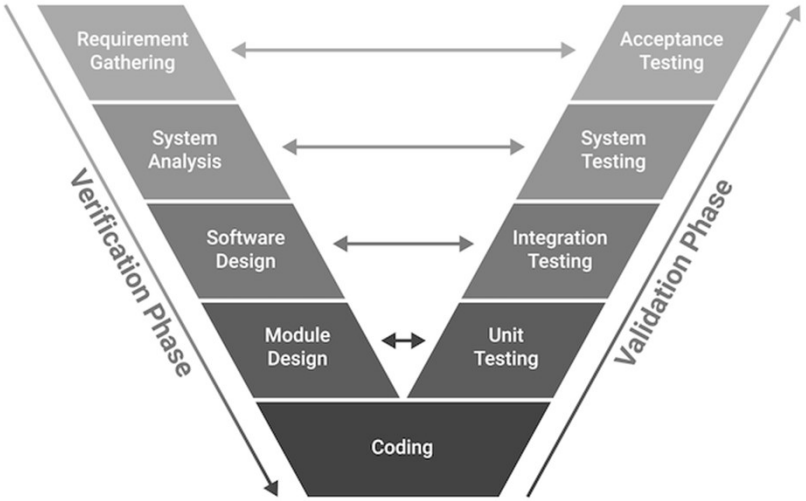
Modele de dezvoltare software

Există diferite abordări de dezvoltare software definite și proiectate care sunt 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 agile, model spiralat, model prototip etc.). Fiecare model urmează un anumit ciclu de viață pentru a asigura succesul în procesul de dezvoltare a software-ului.


Kanban
Iterative&Incremental Model
Scrum
RUP
Spiral
XP
Waterfall
V-model
SEQUENTIAL → EVOLUTIONARY
FORMAL → INFORMAL



Waterfall Model



V Model

Modelul cascadă

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.

Avantaje și dezavantaje în utilizarea modelului cascadă

A

01 Acest model este simplu și ușor de înțeles și utilizat.

02 În acest model, fazele sunt procesate și finalizate una câte una. Fazele nu se suprapun.

03 Modelul cascadă funcționează bine pentru proiecte mai mici unde cerințele sunt foarte bine înțelese.

D

01 Odată ce o aplicație este în faza de testare, este dificil să te întorci și să schimbi ceva care nu a fost bine planificat.

02 Nu este produs niciun software funcțional până târziu în timpul ciclului de viață. Lipsește **produsul minim viabil.**

03 Nu este un model bun pentru proiecte complexe și orientate pe obiecte. Model slab pentru proiecte lungi și aflate în derulare.

Când este recomandat modelul cascadă ?

- Definiția produsului este stabilă.
- Tehnologia este înțeleasă.
- Resursele ample cu expertiza necesară sunt disponibile gratuit
- Proiectul este scurt.

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

Modelul - V

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, iar funcționalitatea lor este testată.

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

Modelul - V

Specificația cerințelor - Nevoile și cerințele clientului sau viitorului utilizator al sistemului sunt adunate, specificate și aprobate.
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).
Specificația programului - Acest pas definește fiecare subsistem, inclusiv sarcina, comportamentul, structura interioară și interfețele cu alte subsisteme.
Codare - Fiecare componentă specificată 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 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.

Avantaje și dezavantaje în utilizarea modelului V

A

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

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

03 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.

D

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

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

03 Foarte rigid și mai puțin flexibil.

Când este recomandat modelul V ?

- Cerințele sunt bine definite, clar documentate și fixate.

- Definiția produsului este stabilă.

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

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.

Spiral Model

Agile Development

Modelul spirală

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.

Modelul spirală

Planificarea cerințelor - Cerințele sunt adunate în timpul fazei de planificare.

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.

Avantaje și dezavantaje în utilizarea modelului spirală

A

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

02 Funcționalități suplimentare pot fi adăugate la o dată ulterioară.

03 Bun pentru proiecte mari și critice pentru misiune. Control puternic de aprobare și documentație.

D

01 Poate fi un model costisitor de utilizat.

02 Analiza riscurilor necesită o expertiză foarte specificată. Succesul proiectului depinde de faza de analiză a riscului.

03 Nu funcționează bine pentru proiecte mai mici.

Când este recomandat modelul spirală ?

- 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

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.

Caracteristici Agile

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 – membrii echipei își raportează unul altuia ceea ce au făcut în ziua precedentă, ce intenționează să facă astăzi și care sunt obstacolele lor.

Avantaje și dezavantaje în utilizarea dezvoltării agile

A

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

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

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

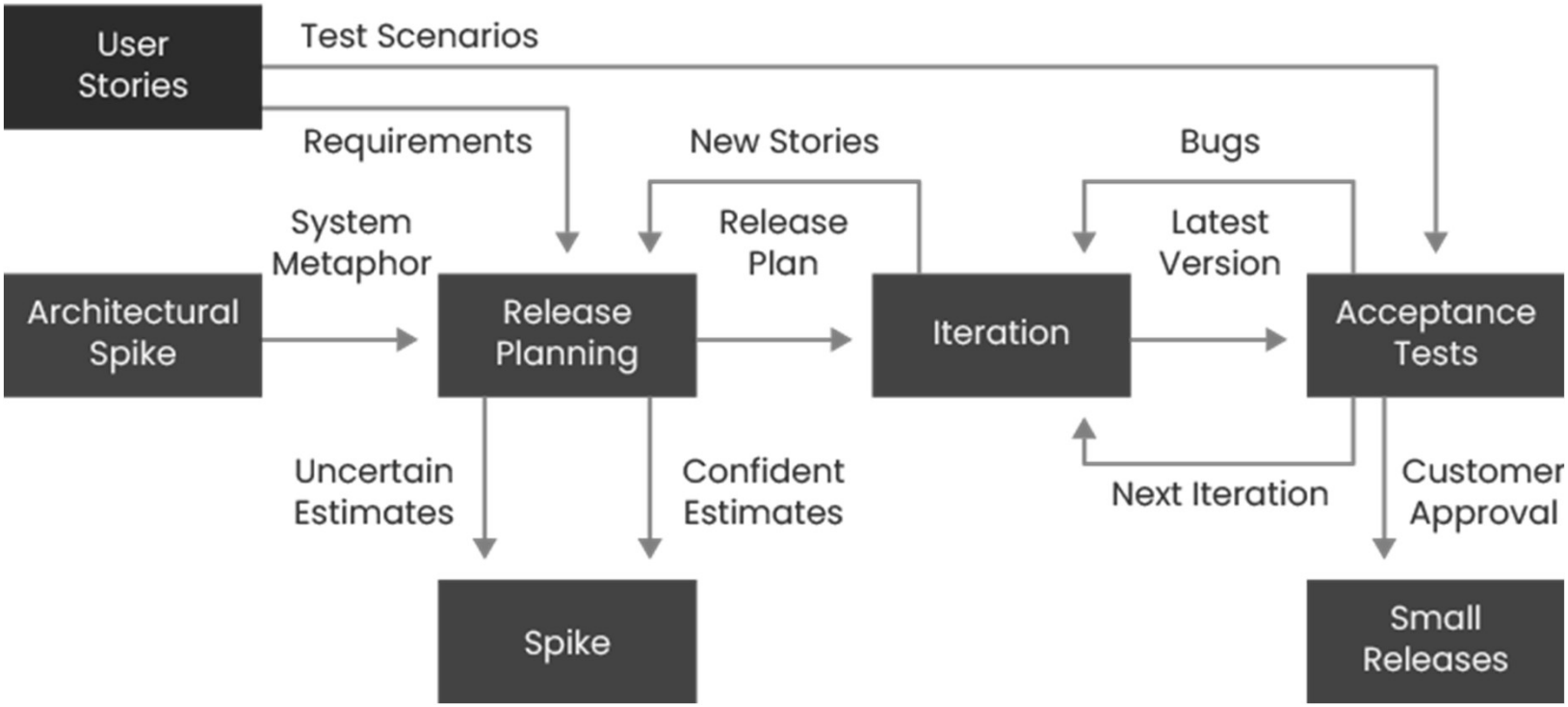
D

01 În cazul unor livrabile software, în special a celor mari, este dificil de evaluat efortul necesar la începutul ciclului de viață al dezvoltării software.

02 Nu se pune accent pe proiectarea și documentarea necesară.

03 Doar programatorii seniori sunt capabili să ia tipul de decizii necesare în timpul procesului de dezvoltare. Prin urmare, nu are loc pentru programatorii începători.

Extreme Programming (XP) Methodology



Extreme Programming (XP) methodology

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”.

Extreme Programming (XP) methodology

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

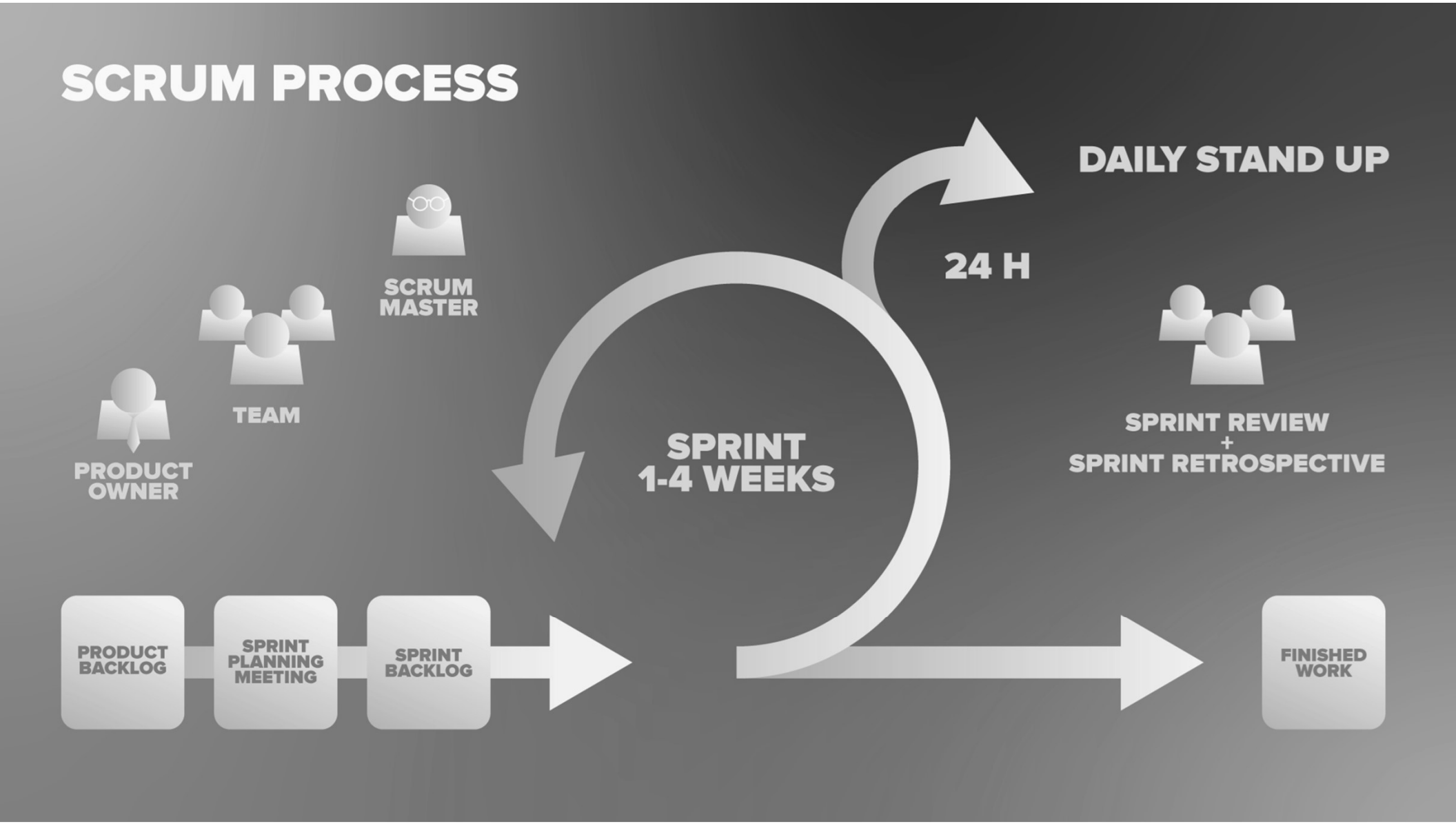
Verificați dacă o funcție funcționează testând-o. XP folosește teste unitare, care sunt teste automate, iar programatorul scrie cât mai multe dintre ele pentru a încerca să spargă codul la care lucrează.

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.

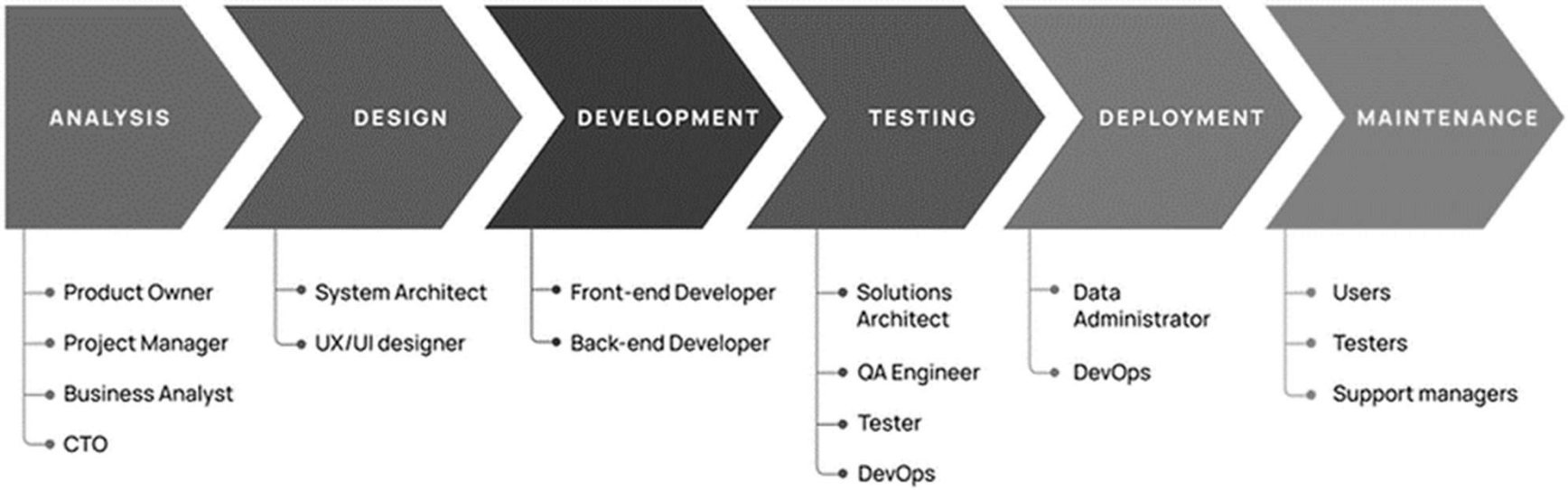

SCRUM PROCESS
DAILY STAND UP
24 H
SCRUM MASTER
TEAM
PRODUCT OWNER
SPRINT REVIEW + SPRINT RETROSPECTIVE
SPRINT 1-4 WEEKS
PRODUCT BACKLOG
SPRINT PLANNING MEETING
SPRINT BACKLOG
FINISHED WORK

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.**

Sarcină practică

Argumentează în baza schemei care este rolul fiecărui membru din echipa de dezvoltare în elaborarea produselor software.


ANALYSIS
DESIGN
DEVELOPMENT
TESTING
DEPLOYMENT
MAINTENANCE
Product Owner
Project Manager
Business Analyst
CTO
System Architect
UX/UI designer
Front-end Developer
Back-end Developer
Solutions Architect
QA Engineer
Tester
DevOps
Data Administrator
DevOps
Users
Testers
Support managers