Când menţionăm noţiunea de dezvoltare software, primul gând pentru majoritatea este programarea. Tocmai de aceea, lecțiile anterioare au fost dedicate familiarizării cu postulatele de bază ale programării. Într-adevăr, dacă intenţionaţi să învăţaţi să creaţi programe de calculator pe cont propriu, programarea este disciplina principală de învăţat. Cu toate acestea, când învăţaţi programarea şi deveniţi inginer software, asta înseamnă că cunoaşteţi şi profesii înrudite, dar nu identice.

Dezvoltarea software profesională implică mult mai mult decât scrierea codului. Pentru început, dezvoltarea profesională implică aproape întotdeauna echipe de dezvoltare. Dezvoltarea software într-o echipă este complet diferită de crearea programelor de calculator în mod independent. Dezvoltarea software profesională constă în numeroase alte operaţiuni, efectuate de angajaţi de diverse profesii. Redactarea documentaţiei, proiectarea, analizarea, testarea, mentenanţa, suportul - acestea sunt doar câteva dintre operaţiunile implicate în dezvoltarea software. Dezvoltarea software profesională se ocupă de o disciplină altfel cunoscută sub numele de inginerie software.

În aceasta şi în următoarele câteva lecţii, ne vom ocupa cu postulatele de bază ale ingineriei software, astfel încât veţi avea ocazia să vă faceţi o imagine mai clară asupra tuturor aspectelor dezvoltării software. Vom începe prin a defini mai detaliat termenul de inginerie software.

Ce este ingineria software?

- Ingineria software este o disciplină de inginerie care se ocupă de toate aspectele dezvoltării produselor software de la primele etape ale creării unei specificaţii software, până la livrare şi mentenanţă. Inginerii software se asigură că un produs software îndeplineşte aşteptările în ceea ce priveşte stabilitatea, viteza, utilizarea, upgrade-ul şi testarea, costul, securitatea şi diverşi alţi factori. Pentru a realiza aşa ceva, produsul software trebuie să aibă 4 caracteristici de bază: uşurinţa de mentenanță, fiabilitate şi securitate, eficienţă şi

acceptabilitate.

- **Uşurinţa de mentenanţă**
  Toate produsele software necesită îmbunătăţire/upgrade constant. Acest lucru este dictat de circumstanţele afacerii, concurenţă, cerinţele clienţilor şi mulţi alţi factori. Prin urmare, orice produs software ar trebui proiectat astfel încât să poată fi îmbunătăţit cu uşurinţă. Posibilitatea de mentenanţă uşoară este un imperativ care reduce costurile şi creşte disponibilitatea în timpul exploatării unui produs software.
- **Fiabilitatea şi securitatea**
  Fiecare produs software ar trebui să fie fiabil şi sigur. Software-ul fiabil funcţionează conform aşteptărilor. Un astfel de software nu provoacă daune fizice şi economice în cazul unei eventuale eşuări. Software-ul securizat nu îşi expune în niciun fel resursele utilizatorilor maliţioşi (hackerilor).
- **Eficienţa**
  Eficienţa produselor software se reflectă în utilizarea raţională a resurselor sistemului, cum ar fi memoria de lucru şi procesorul. Fiecare produs software ar trebui, cu cantitatea minimă de resurse de sistem, să realizeze operaţiunea pe care utilizatorul o aşteaptă de la el în cel mai rapid mod.
- **Acceptabilitatea**
  În cele din urmă, fiecare produs software trebuie să fie acceptabil pentru utilizatorii săi. Acceptabilitatea este un termen care se referă la înţelegere, uşurinţă în utilizare şi compatibilitate. Fără nici una dintre caracteristicile menţionate, se poate spune că programul nu este acceptabil pentru utilizatorii finali.

De ce avem nevoie de ingineria software?

Pentru a înţelege motivele existenţei ingineriei software, la început este bine să ne întoarcem în trecut şi la vremea când a fost creat acest termen. Termenul de inginerie software (*Software engineering* în engleză) a fost inventat în anul 1968 la prima conferinţă privind ingineria software-ului organizată de Alianţa Tratatului Atlanticului de Nord (ONAT/NATO). La acea conferinţă au fost definite liniile directoare

de bază şi cele mai bune practici în ceea ce priveşte dezvoltarea software. Motivul direct al acestei conferinţe a fost aşa-numita *criză a software-urilor* care a avut loc la începutul anilor şaizeci ai secolului trecut.

Criza software (*Software crisis* în engleză) este un termen inventat la conferinţa despre ingineria software-ului tocmai menţionată, în anul 1968. Este vorba de un termen care descrie situaţia în care s-a aflat dezvoltarea profesională a software-ului la începutul anilor şaizeci. Şi anume, din cauza progresului mare al hardware-ului computerului, s-a înregistrat o creştere uriaşă a complexităţii programelor de calculator care nu au reuşit să ţină pasul cu progresul hardware. Ca urmare, au apărut numeroase probleme cum ar fi:

- depăşirea bugetului,
- depăşirea termenelor limită,
- calitate slabă şi optimizare slabă,
- securitate inadecvată,
- decese cauzate de erori software.

Pe scurt, ingineria software a apărut ca răspuns la problemele dezvoltării unor sisteme software mari care au fost dezvoltate cu întârziere, depăşind bugetele şi nereuşind să îndeplinească aşteptările utilizatorilor în ceea ce priveşte funcţionalitatea, performanţa şi fiabilitatea.

OS/360 – un exemplu inadecvat de depăşire a bugetelor şi a termenelor limită

De-a lungul istoriei dezvoltării software, a existat un număr mare de proiecte software care şi-au depăşit bugetul iniţial şi termenele limită stabilite. Cu toate acestea, unul dintre cele mai interesante astfel de exemple este dezvoltarea unui sistem de operare numit OS/360 de către IBM. Este vorba de unul dintre cele mai scumpe produse software din toate timpurile.

Dezvoltarea sistemului de operare OS/360 a început în anul 1964. Bugetul său iniţial a fost de 25 de milioane de dolari, ceea ce a fost

suficient pentru o echipă de dezvoltare de 60 de programatori. În octombrie 1965, echipa de dezvoltare a crescut la 150 de programatori, iar termenul de livrare a fost prelungit cu 6 luni. În anul următor (1966), echipa de dezvoltare a fost formată din peste 1000 de programatori. OS/360 a fost finalizat în cele din urmă cu mai mult de un an de întârziere şi la un cost care a depăşit 5 miliarde de dolari. Este vorba de un preț care a fost de două ori mai mare decât veniturile anuale ale IBM.

Totuşi, această poveste nu a avut un final tragic. Sistemul de operare OS/360 a devenit în cele din urmă un produs de succes. În mai puţin de 5 ani, IBM a reuşit să returneze din nou banii investiţi, în timp ce în următoarele două decenii, mai mult de jumătate din veniturile IBM au provenit din produse legate de acest sistem de operare. Chiar şi aşa, acesta este un exemplu izolat de produs de succes după toate întârzierile şi depăşirile de buget în timpul dezvoltării. În majoritatea situaţiilor similare, produsele software cu probleme identice nici nu ar fi văzut lumina zilei.

Therac-25 – un exemplu de eroare software care a costat vieţi

Depăşirea termenelor şi a bugetelor sunt probleme serioase din punctul de vedere al companiilor şi clienţilor. Totuşi, în unele situaţii, software-ul poate fi de vină pentru situaţii mult mai grave, cum ar fi pierderea de vieţi ale oamenilor. Maşina Therac-25 şi software-ul care a rulat acea mașină vor rămâne înscrişi în istorie.

Therac-25 a fost o mașină pentru radiaţii terapeutice, adică o maşină care folosea raze X pentru a distruge celulele canceroase din corpul uman. Cu toate acestea, din cauza unei erori în software-ul care controla acea mașină, patru persoane au murit. O eroare software (*Bug* în engleză) a făcut ca Therac-25 să expună anumiţi pacienţi la radiaţii de 100 de ori mai mari decât s-a prevăzut.

Criza software este un termen care a fost inventat în anii şaizeci ai secolului trecut, dar de-a lungul timpului a devenit un sinonim universal pentru toate problemele care apar în timpul dezvoltării software. Prin urmare, este un termen relevant şi în ziua de astăzi, ori de câte ori trebuie să se definească dezvoltarea software care nu decurge conform planului.

În ce fel rezolvă ingineria software criza software?

- Ingineria software rezolvă criza software prin introducerea diferitor concepte în lumea dezvoltării software. Acestea sunt: procese de dezvoltare, profesionalism, instrumente software (CASE), paradigme, metodologii, metode formale.
- **Procesele de dezvoltare**
  Procesele de dezvoltare sunt activităţi utilizate în procesul de creare a unui produs software. Cele mai frecvente activităţi sunt următoarele: colectarea cerinţelor utilizatorilor, analiza cerinţelor, proiectarea sistemului, dezvoltarea, testarea, implementarea, mentenanţa...
- **Profesionalismul**
  Ingineria software este o disciplină profesională, adică o profesie. Aceasta introduce diferite forme de educaţie formală şi informală, certificare, etica profesională şi responsabilitate în povestea dezvoltării software. Toate acestea au ca rezultat o abordare mai serioasă a dezvoltării software profesionale.
- **Instrumentele software (CASE)**
  CASE este abrevierea de la *Computer Aided Software Engineering*. Este vorba de un grup de instrumente software care susţin toate etapele dezvoltării software. Diferite tipuri de programe software care pot fi numite instrumente CASE sunt: programele pentru desenarea diagramelor, programele pentru colectarea cerinţelor, programele pentru generarea rapoartelor, programele pentru analiza structurii software, programele pentru crearea documentaţiei, programele pentru generarea codurilor, programele de testare...
- **Paradigmele**
  Paradigma este un termen care defineşte etapele precise

pentru rezolvarea unei probleme de programare. Paradigmele pot fi, de asemenea, observate ca funcţionalităţi pe care le deţin limbajele de programare. Cele mai cunoscute paradigme sunt: programarea orientată pe obiecte, programarea procedurală, programarea bazată pe evenimente...

- **Metodologiile**
  Metodologia determină modul în care este gestionată întreaga dezvoltare a produsului software. Aceasta defineşte procesele care alcătuiesc munca la un proiect de dezvoltare software. Cele mai cunoscute metodologii sunt: Agile, Waterfall, Lean...
- **Metodele formale**
  Metodele formale din lumea dezvoltării software se referă la diferite modele matematice utilizate pentru a rezolva probleme în timpul colectării cerinţelor, dezvoltării specificaţiilor şi proiectării sistemului software.

Procesul de dezvoltare software

Procesul de dezvoltare software este o abordare sistematică care descrie o serie de activităţi care se desfăşoară în timpul dezvoltării unui produs software. Procesul de dezvoltare software a fost deja menţionat în rândurile anterioare în care am spus că aceasta este una dintre abordările de bază prin care ingineria software rezolvă criza software, prin definirea clară a etapelor care alcătuiesc un proiect pe care se realizează dezvoltarea unui produs software.

Există patru activităţi de bază care alcătuiesc procesul de dezvoltare software:

- **specificaţia software** – o activitate în timpul căreia clienţii şi echipa de dezvoltare definesc caracteristicile software-ului care trebuie creat,
- **dezvoltarea software** – o activitate în timpul căreia se realizează proiectarea şi programarea unui produs software,
- **validarea software** – o etapă în timpul căreia se verifică dacă programul îndeplineşte cerinţele utilizatorului,

- **evoluţia software** – etapa în care software-ul este modificat pentru a răspunde noilor cerinţe ale utilizatorilor şi schimbărilor pieţei.

Cele patru etape sau activităţi sus-menţionate sunt considerate etape de bază atunci când vine vorba de dezvoltarea software-ului. Cu alte cuvinte, fiecare proces de dezvoltare software, într-o formă oarecare, trebuie să aibă activităţile tocmai menţionate. Astfel de activităţi sunt uneori denumite diferit sau împărţite într-un număr de activităţi mai mici. Veţi putea citi exact asta în următorul capitol, care va aborda modelele de dezvoltare software.

Modele de dezvoltare software

O reprezentare simplificată a procesului software este altfel numită model de dezvoltare software. Modelul defineşte etapele care se parcurg în timpul dezvoltării, precum şi ordinea acestora. Modelul de dezvoltare poate fi observat ca o viziune specifică a procesului de dezvoltare, respectiv ca o vedere a procesului de dezvoltare dintr-o anumită perspectivă.

Cele două modele de bază ale dezvoltării software sunt următoarele:

- **modelul de dezvoltare în cascadă (waterfall)** - un model de dezvoltare care altfel este numit secvenţial,
- **modelul de dezvoltare agilă** – un model de dezvoltare care poate fi caracterizat ca incremental şi iterativ.

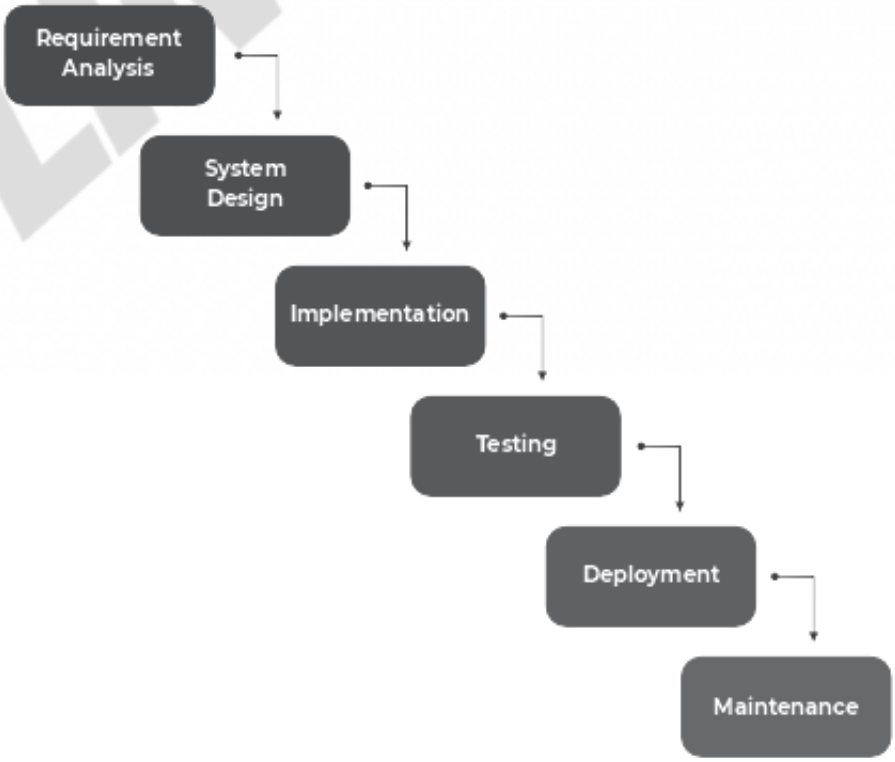
Rândurile următoare vor fi consacrate unei analize detaliate a celor două modele de dezvoltare tocmai menţionate.

Modelul de dezvoltare în cascadă (waterfall)

Waterfall este primul model de dezvoltare utilizat pe scară largă. Ca termen a apărut în anul 1956. Poate fi observat ca o realizare clasică a ideii pe baza căreia munca de dezvoltare software este împărţită într-un număr de etape interdependente. Astfel de etape, în cazul modelului de dezvoltare în cascadă, sunt următoarele:

- analiza cerinţelor,
- proiectarea sistemului,
- implementarea,
- testarea,
- livrarea,
- mentenanţa.

Caracteristica principală a modelului de dezvoltare în cascadă (waterfall) este liniaritatea, adică execuţia secvenţială a etapelor tocmai menţionate. Aceasta înseamnă practic că o etapă trebuie să fie complet finalizată pentru a trece la următoarea etapă. Imaginea 7.1. ilustrează acest lucru.



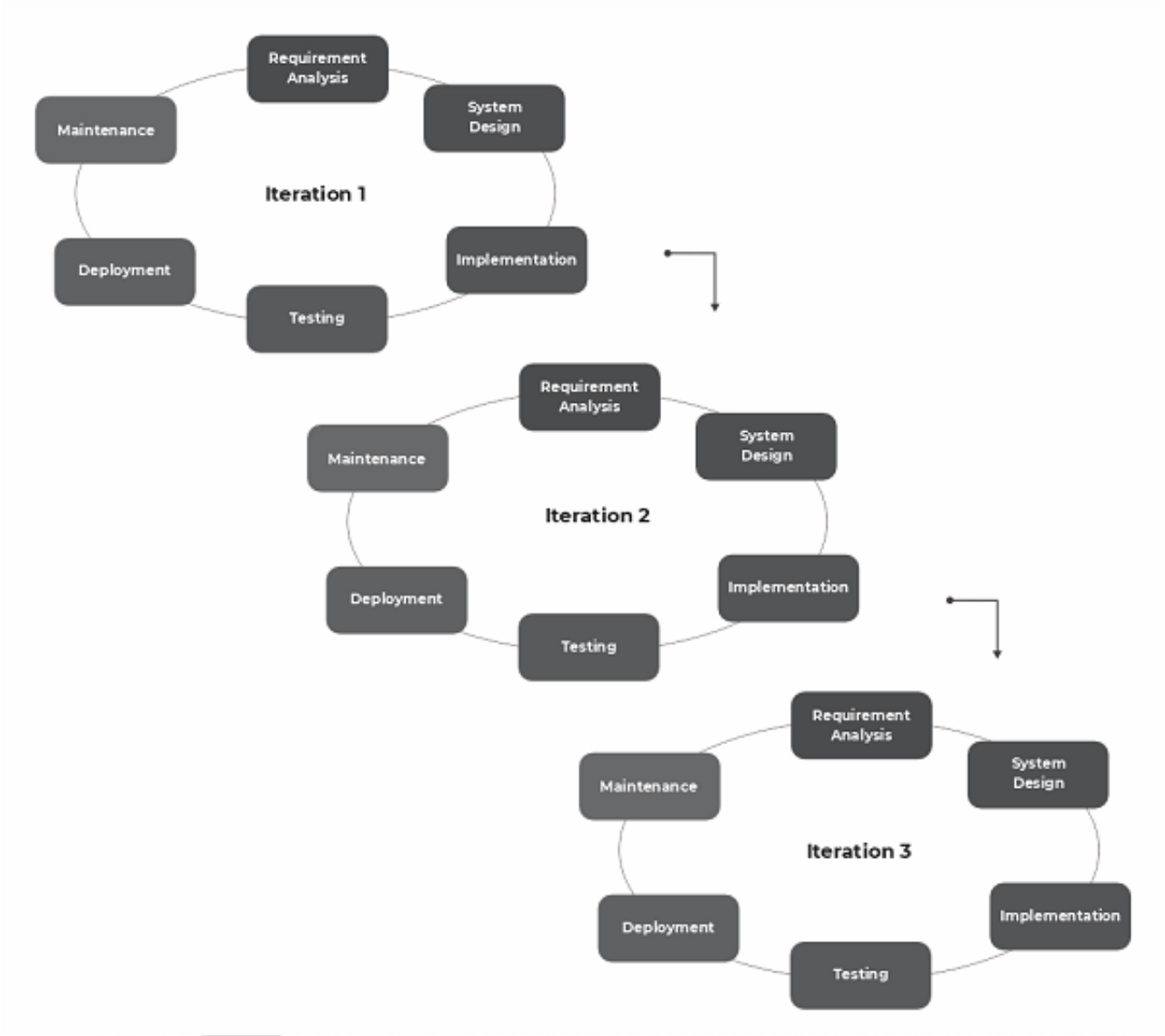
Imaginea 7.1. Modelul de dezvoltare în cascadă (waterfall)

În imaginea 7.1. puteţi vedea în cel mai bun mod de ce acest model de dezvoltare este numit cascadă (waterfall). Lucrul la proiect se desfăşoară din etapă în etapă şi, la fel ca o cascadă, fluxul poate avea o singură direcţie. Aceasta înseamnă că numai după finalizarea etapei anterioare poate începe următoarea etapă de dezvoltare.

Waterfall este un model de dezvoltare care este adesea caracterizat ca fiind prea rigid, cu o bază strictă şi nepractic pentru dezvoltarea software-ului modern de astăzi. Cel mai mare dezavantaj este atribuit flexibilităţii sale reduse, ceea ce înseamnă că produsul software trebuie să fie complet finalizat pentru a fi livrat. O astfel de caracteristică rezultă direct din faptul că acest model presupune implementarea secvenţială a etapelor. Implementarea secvenţială a etapelor complică semnificativ modificarea specificaţiei originale, ceea ce face ca produsul software să fie dificil de întreţinut. Din această cauză, de-a lungul timpului, modelul de dezvoltare agilă a devenit foarte popular, ceea ce oferă o alternativă la modelul tocmai descris, cu o organizare diferită a etapelor procesului de dezvoltare.

Modelul de dezvoltare agilă

Modelul de dezvoltare agilă implică etape identice ca modelul cascadă (waterfall) tocmai descris. Totuşi, organizarea unor astfel de etape în dezvoltarea agilă este semnificativ diferită (imaginea 7.2.).



Imaginea 7.2. Modelul de dezvoltare agilă

În imaginea 7.2. puteţi vedea cum arată un model simplificat de dezvoltare agilă. Toate etapele, cum ar fi analiza cerinţelor, proiectarea, construcţia şi altele, există ca şi până acum. Cu toate acestea, în dezvoltarea agilă, acestea sunt organizate într-un mod uşor diferit.

Se spune că dezvoltarea agilă este incrementală şi iterativă. Pentru început, întregul proces de dezvoltare a unui program este împărţit într-un număr mare de unităţi mai mici, numite incremente. Echipa de

dezvoltare dezvoltă produsul software în părţi, increment cu increment, adică funcţionalitate după funcţionalitate.

Iteraţiile sunt perioade scurte de timp în care echipa de dezvoltare lucrează la realizarea unei funcţionalităţi. În timpul unei iteraţii se trece prin toate etapele de dezvoltare, de la planificare, implementare până la testare. Astfel, fiecare iteraţie are ca rezultat un software complet funcţional, a cărui revizuire duce la următoarea iteraţie. Se poate spune că produsul iteraţiei este un increment. În imaginea 7.2. puteţi vedea un exemplu de trei iteraţii.

Modelul de dezvoltare agilă aduce numeroase avantaje, cum ar fi:

- **adaptabilitatea la schimbări** – având în vedere că dezvoltarea software este împărţită într-un număr mai mare de unităţi mai mici, iar dezvoltarea are loc în etape, capacitatea de adaptare la schimbări este mult mai bună;
- **livrarea rapidă a noilor funcţionalităţi** – spre deosebire de modelul în cascadă, în cazul dezvoltării agile, perioada de aşteptare a livrării unui produs funcţional este mult mai scurtă, deoarece se încearcă **lansarea unei versiuni a produsului cu un set de bază de funcţionalităţi încă de la început şi în iteraţiile ulterioare astfel de funcționalităţi sunt îmbunătăţite şi sunt adăugate altele noi;**
- **corectarea simplă a erorilor** – datorită dezvoltării incrementale, potențialele erori au un efect dăunător mult mai mic asupra funcţionării sistemului şi sunt mai simplu de detectat şi de eliminat;
- **planificarea simplificată** – planificarea individuală a funcţionalităţilor mai mici este mult mai simplă decât planificarea unui produs software complet.