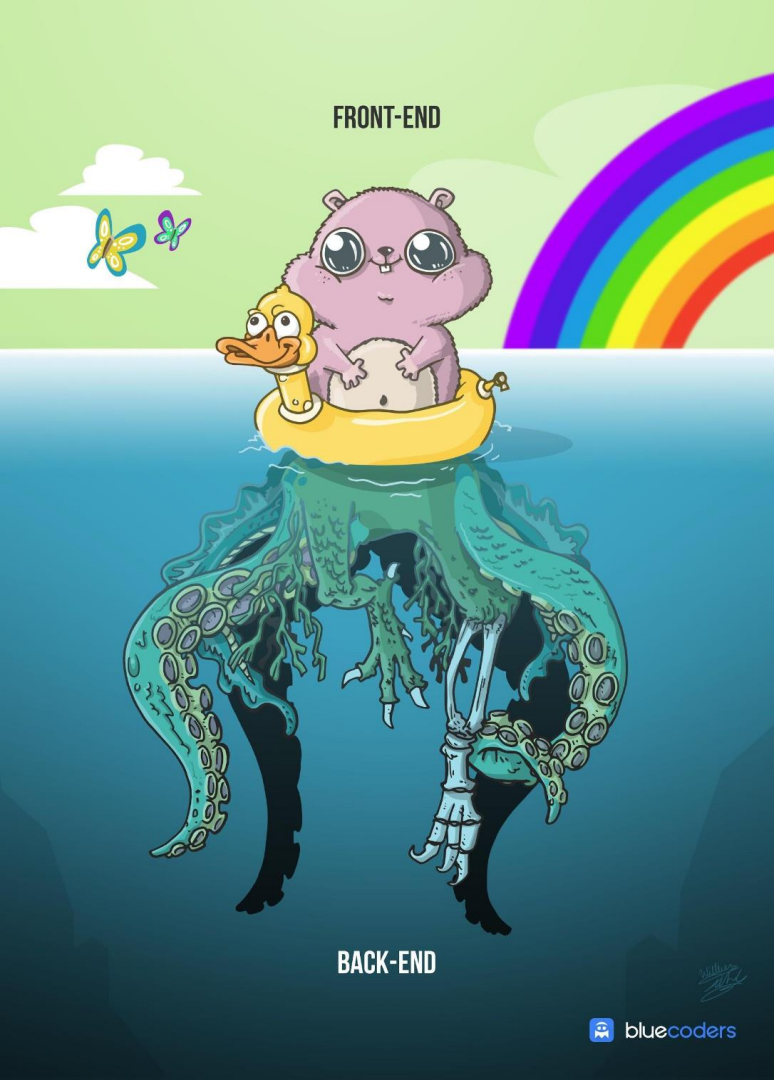


Now you clearly

wantsome
The friendly IT Academy

Frontend lessons

Welcome future *frontenders!*

wantsome
The friendly IT Academy

Saptamana 1

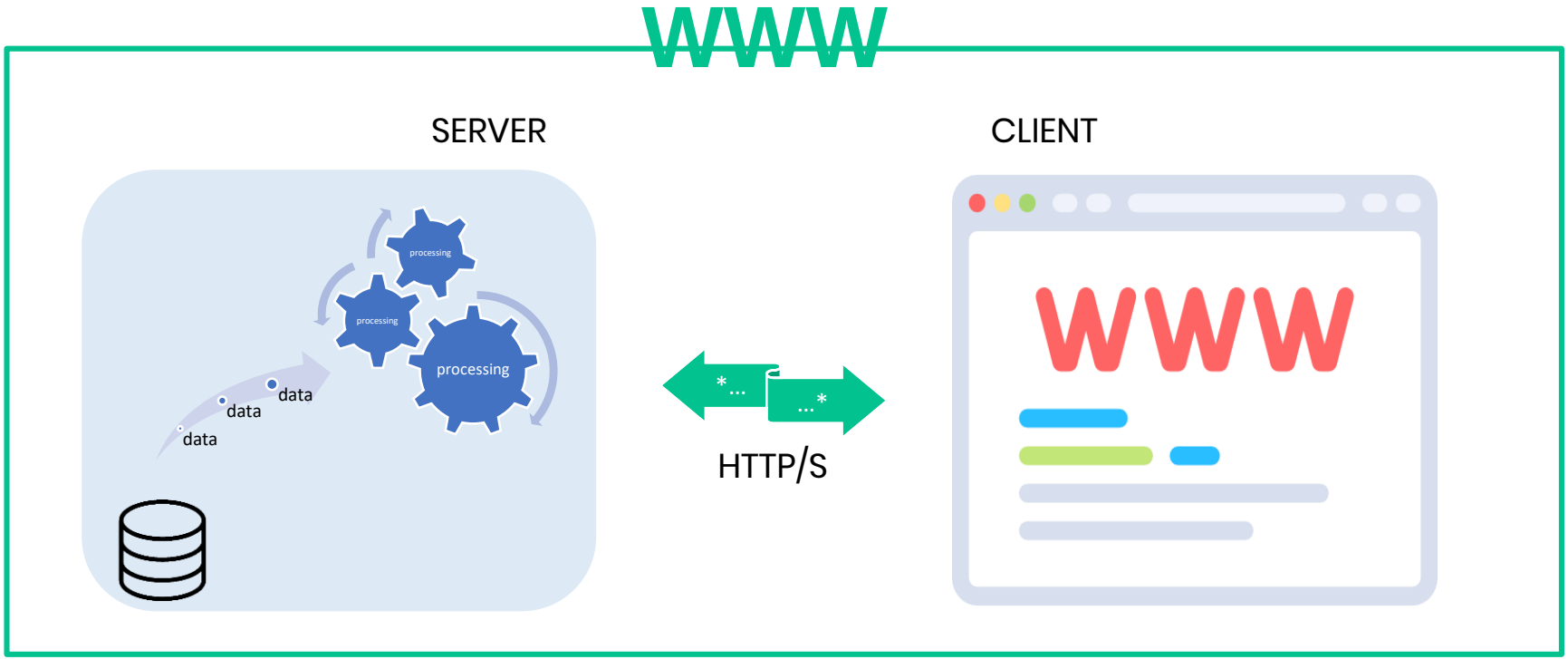
Partea 1

Programare Front-End

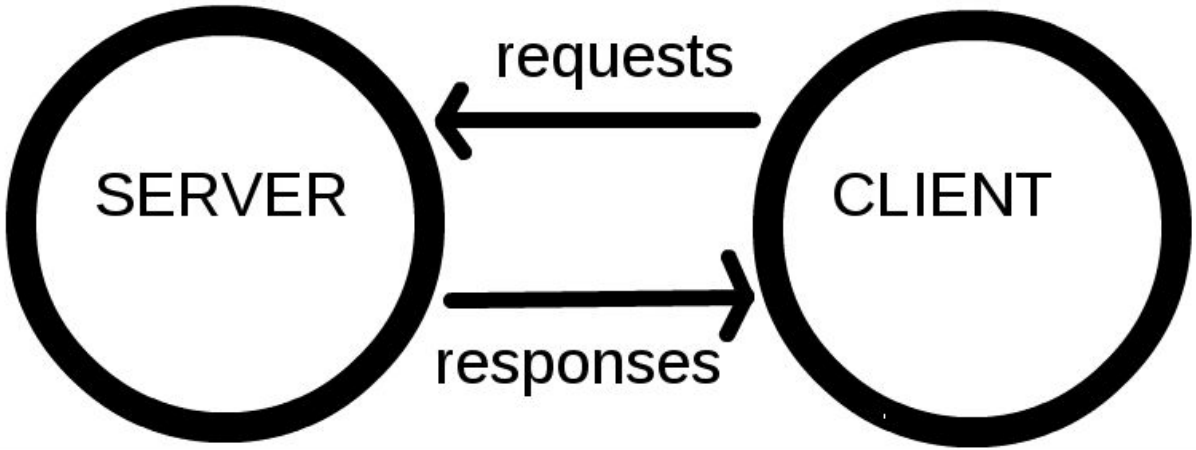
I. What about „Front-End Development”?

1. Web Apps Architecture Overview

Server - Retea - Client



Server - Retea - Client



Server - Retea - Client

Notiuni:

- **World Wide Web (WWW)**: Spatiu informatic in cadrul caruia diferite tipuri de documente si resurse sunt identificate de URLs (Uniform Resource Locators) accesibile prin intermediul *Internet*-ului

- **Server**: Calculator care stocheaza pagini web, *site*-uri sau aplicatii, care opereaza in mod continuu in cadrul unei retele, asteptand solicitari (cereri) din partea altor dispozitive din retea (*clienti*); Cand un dispozitiv client doreste sa acceseze o pagina web, o copie a paginii web este descarcata de pe server pe masina clientului pentru a putea fi afisata in *browser*-ul web al utilizatorului

  BACK-END

- **Client:** Dispozitiv al unui utilizator web conectat la internet (de exemplu, calculatorul tau conectat la *Wi-Fi*, sau telefonul tau conectat la reteaua de telefonie mobila) + softul de accesare a paginilor web disponibil pe acest dispozitiv (de obicei un *browser web*: ex. Firefox, Chrome, Internet Explorer)

  FRONT-END

Server - Retea - Client

Notiuni:

- **Retea:** Canalul principal de comunicare intre ***Client - Server***

- **TCP/IP**: *Transmission Control Protocol* si *Internet Protocol* sunt protocoale de retea care definesc felul in care datele trebuie sa circule la nivel **WEB**; Primul controleaza felul in care se transmit pachetele ce alcatuiesc mesajele transmise intre clienti si servere, iar cel de-al doilea se ocupa cu asignarea de adrese pentru clienti si *server*-e astfel incat sa se stie cine si cui trimite mesaje

- **DNS:** Server de nume de domeniu - practic, o “agenda” a *site*-urilor existente pe *web*; Atunci cand se introduce adresa sub forma de nume a unui site in bara de navigare a *browser*-ului, se verifica numele de domeniu pentru a gasi adresa reala (**IP**) a *site*-ului, pentru a-l putea prelua; *Browser*-ul trebuie sa afle pe ce *server* se afla *site*-ul, astfel incat sa poata trimite mesaje de tip **HTTP** la locatia potrivita

- **HTTP:** Hypertext Transfer Protocol defineste un limbaj de comunicare folosit atat de **client**, cat si de **server**; Este construit peste **TCP/IP**

Server - Retea - Client

Notiuni:

- **Fisiere componente**: Un *site web* este alcatuit din mai multe fisiere diferite; Avem doua tipuri principale:
    - **Fișiere cod:** *Site*-urile *web* sunt construite in principal din **HTML**, **CSS** și **JavaScript**, dar mai tarziu vom afla si despre alte tipuri de tehnologii
    - **Resurse**: Fisiere media precum imagini, muzica, video, documente Word, fisiere PDF, etc...

Ce se intampla mai exact ?

1. *Browser*-ul comunica cu serverul *DNS* pentru a gasi adresa reala a serverului unde se afla gazduit *site*-ul web

1. *Browser*-ul trimite un mesaj de tip *HTTP*, cerandu-i acestuia o copie a documentului principal care sta la baza site-ului (**index.html**). Acest mesaj, impreuna cu toate celelalte date trimise intre server si client sunt transmise prin intermediul internetului, utilizand *protocolul TCP/IP*

1. Daca *server*-ul accepta cererea de la client, acesta raspunde cu un mesaj care are ca si cod de stare "**200 OK**", iar mai apoi incepe sa trimita fisiere catre *browser* sub forma a mai multor pachete de date de dimensiuni mai mici

1. *Browser*-ul stie sa asambleze aceste pachete mici de date si sa alcatuiasca *site*-ul *web* care este afisat in fereastra sa

Referinta: https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/How_the_Web_works

2. UI / UX

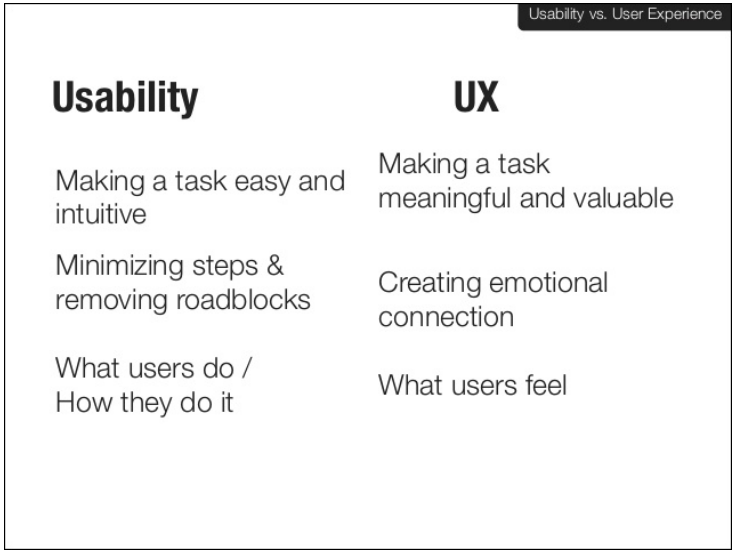
User Interface, User Experience

Notiuni:

- **UI (User Interface)**: Interfata aplicatiei *web*, cea pe care o vede si cu care interactioneaza utilizatorul; este construita si stilizata (colorata, aranjata) cu ajutorul **HTML** si **CSS,** iar partea de interactiune si functionalitate este definita si programata cu ajutorul limbajului de programare **JavaScript**

- **UX (User Experience)**: Experienta utilizatorului; Defineste felul in care utilizatorii se “simt” in momentul in care interactioneaza cu interfata unei aplicatii

User Experience vs Usability

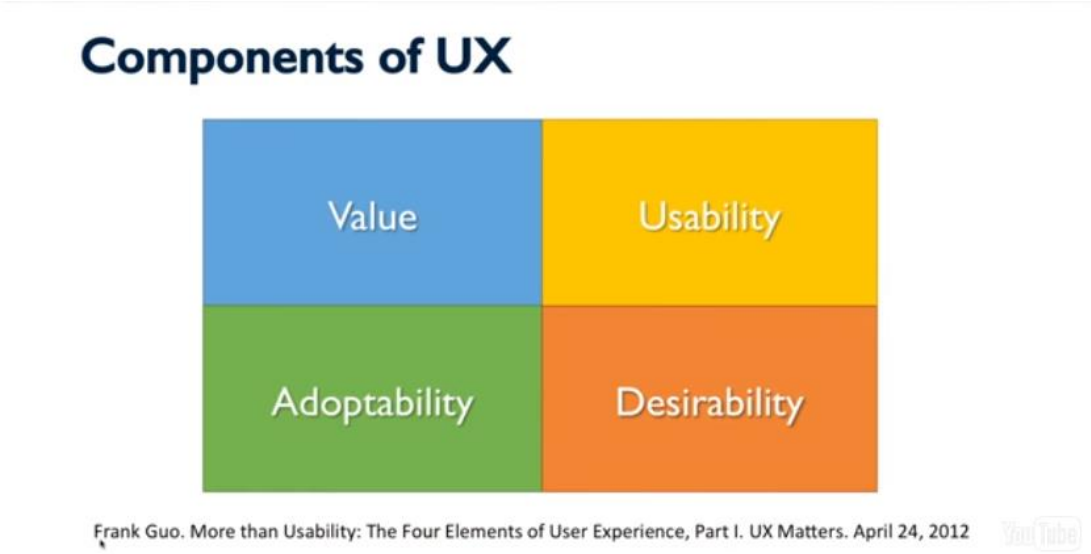


User Experience vs Usability

- **User Experience:** se refera la sentimentele si atitudinea subiectiva a unei persoane legata de utilizarea unui produs anume; Include scopul, *brand*-ul produsului, asteptarile psihologice, emotia in sine

- **Usability:** Un indicator foarte important al calitatii produselor interactive; Se refera la gradul de eficienta al produselor, usurinta de utilizare si invatare, erori putine si nivelul de satisfactie pe care il ofera; Este in primul rand despre partea functionala a produselor, sau pe scurt: "*Don't make me think*" - intuitivitate

User Experience include Usability... si nu numai



Frank Guo. More than Usability: The Four Elements of User Experience, Part I. UX Matters. April 24, 2012

User Experience vs Usability

- **Usability**: “Cat de usor este sa ducem la bun sfarsit un ‘task’ ?”
- **Adaptability**: “Vor incepe oamenii sa foloseasca produsul ?”
- **Desirability**: “Este experienta per ansamblu placuta si captivanta ?”
- **Value**: “Aduce acest produs valoare utilizatorilor ?”

Referinta: https://www.mockplus.com/blog/post/difference-between-usability-and-user-experience

3. Development workflow overview and methodologies

Cum comunicam ?

Cum ne organizam ?

Ce metode folosim ?

Metode de lucru - **SDLC**s

- Exista o diversitate destul de mare pentru metodele de lucru si management al proiectelor in industria IT

- Acestea se pot aplica in forme diferite in functie de tipul de business

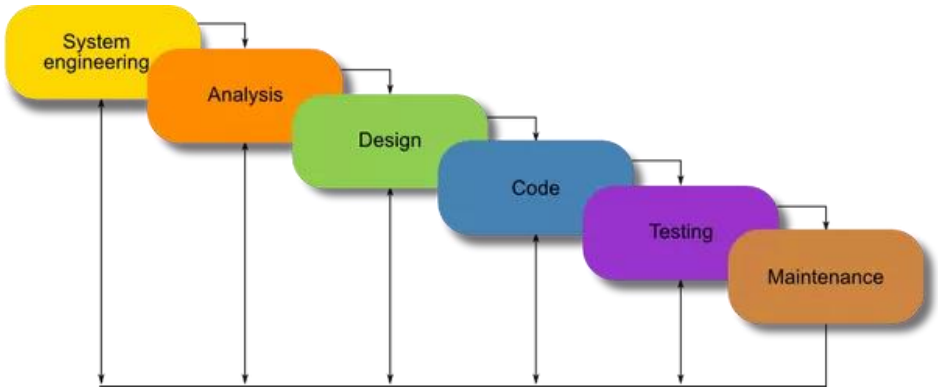
- Printre cele mai intalnite modele de procese pentru dezvoltarea produselor software se enumera:

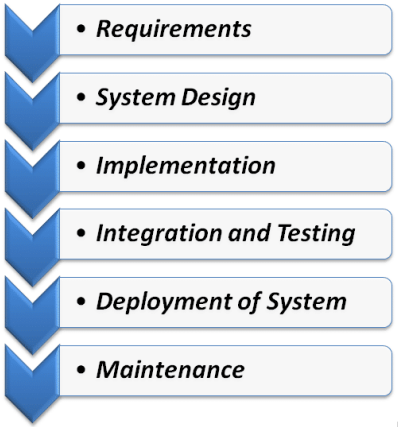
    - **Waterfall**
    - **Agile**
    - **Scrum**
    - **KanBan**

- Aceste modele mai pot fi gasite si in asociere cu denumirea de **SDLC (Software Devopment Life Cycle)**

Waterfall

sau **Dezvoltare in cascada** descrie o metoda in cadrul careia multiple faze ale dezvoltarii unui produs software sunt parcurse liniar, secvential; Fiecare faza trebuie dusa la bun in sfarsit inainte ca urmatoarea sa inceapa deoarece rezultatul primeia in cauza serveste ca baza pentru cea care urmeaza, iar progresul se desfasoara “in jos”, analogic cu felul in care curge o cascada (de aici si numele)



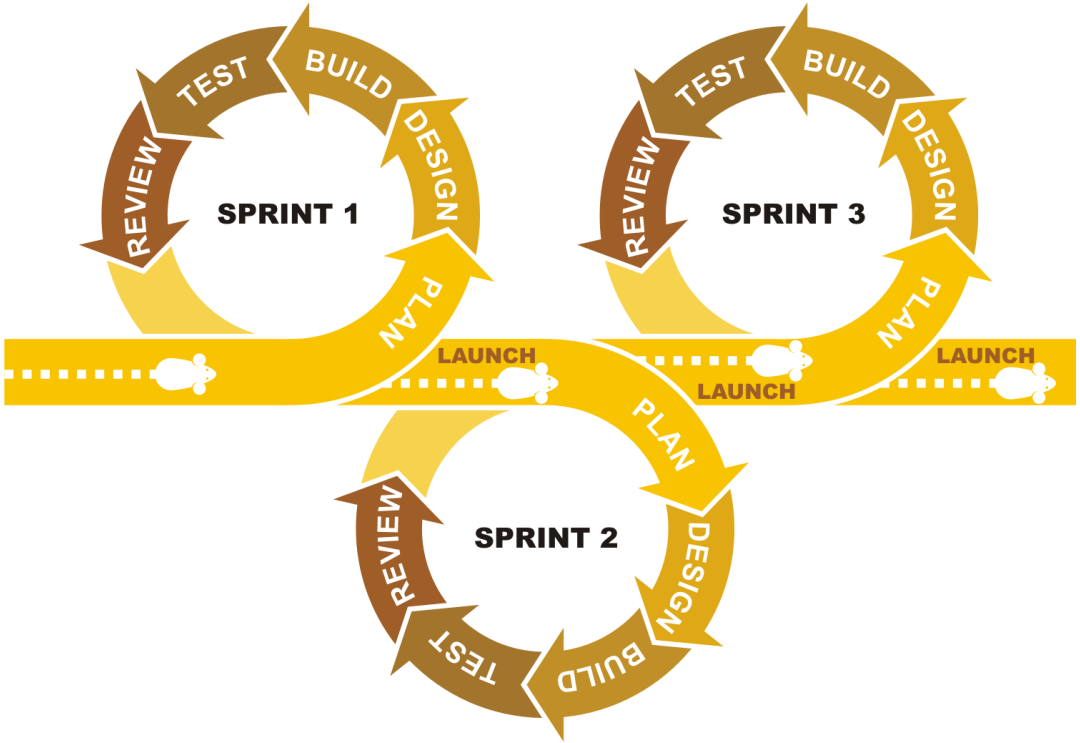


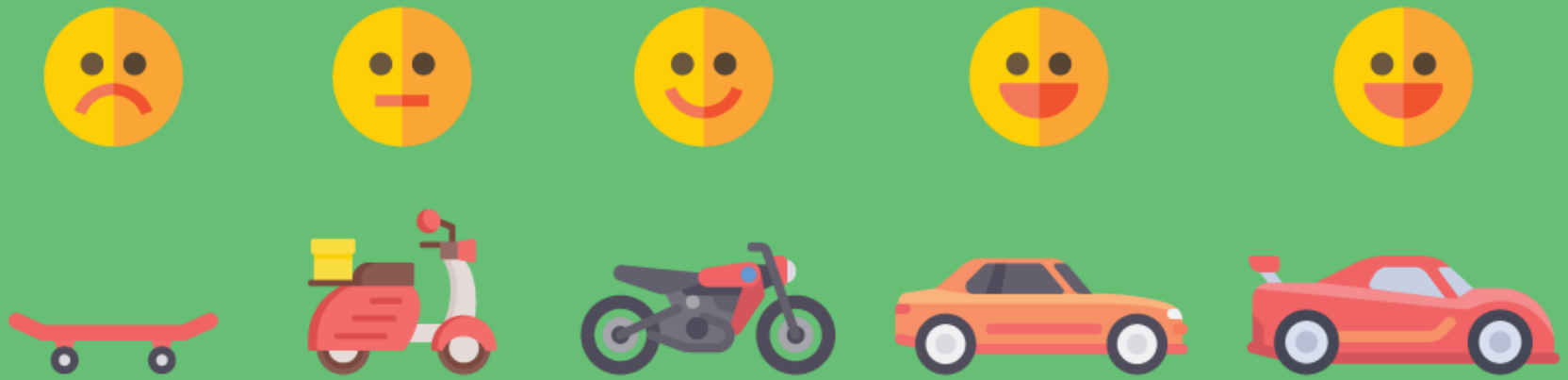
Agile

este o metoda de dezvoltare iterativa care pune pret pe adaptabilitate (in functie de proiect) si satisfactia clientului prin livrarea rapida de software functional. Metodele agile impart produsul intreg in unitati functionale de dimensiuni mai mici. Aceste unitati sunt livrate in interatii. Fiecare iteratie deobicei variaza ca si durata intre 2 - 3 saptamani (nu neaparat). Fiecare iteratie implica mai multe micro-echipe care lucreaza simultan in diferite arii ale proiectului:

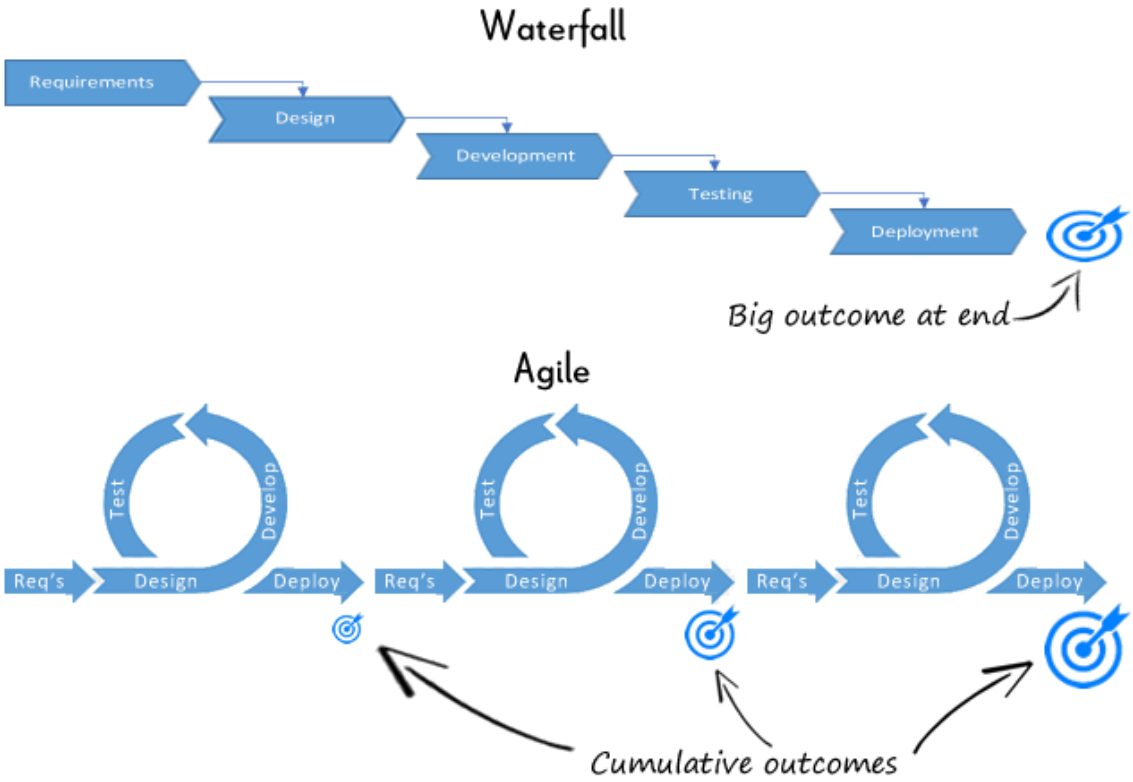
- Planning
- Requirements Analysis
- Design
- Coding
- Unit Testing and
- Acceptance Testing

AGILE METHODOLOGY





Waterfall vs Agile



Principii Agile

1. **“Individuals and interactions”** - motivatia si dezvoltarea personala este la fel precum interactiunea si colaborarea cu ceilalti membri ai echipei

1. **“Working software”** - prezentarea iterativa (**demo**) de software functional este considerata cea mai buna metoda de comunicare cu clientul pentru a-i intelege cerintele, in detrimentul dezvoltarii bazata pe o documentatie foarte amanuntita

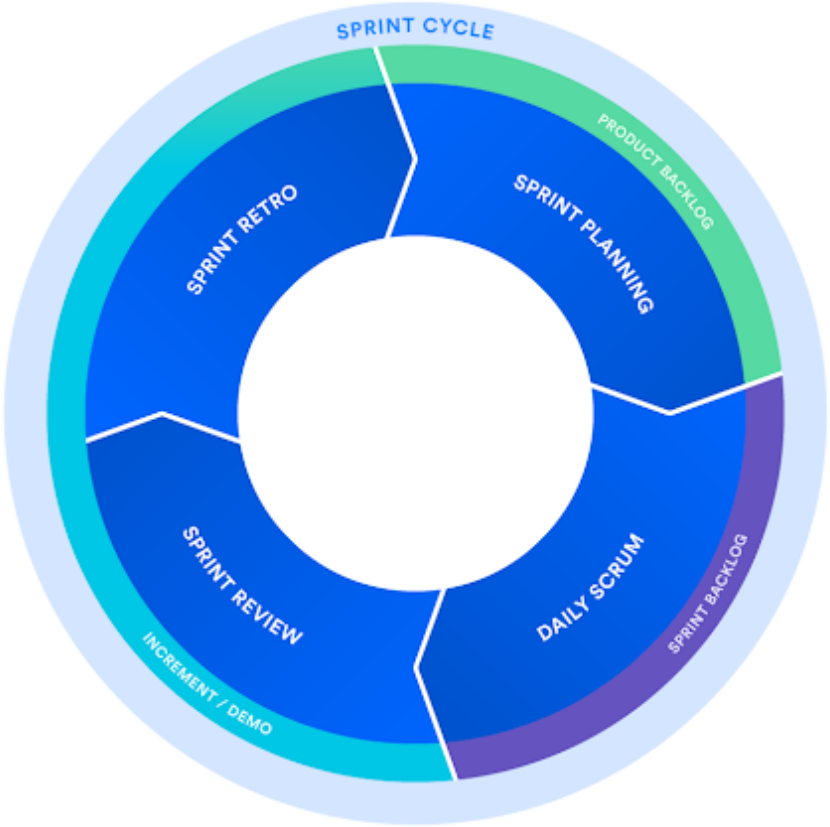
1. **“Customer collaboration”** - deoarece nu toate specificatiile si cerintele proiectului sunt valabile de la inceputul proiectului, comunicarea continua si transparenta cu clientul este foarte importanta pentru progres si pentru a obtine informatiile necesare

1. **“Responding to change”** - pregatirea unui plan amanuntit este inlocuita cu adaptabilitatea la schimbari rapide pe parcursul dezvoltarii continue a unui proiect

SCRUM

este un *framework* care ajuta echipele sa lucreze impreuna, facand analogie la echipele de rugby, de unde vine si numele. *SCRUM* incurajeaza echipele sa invete prin intermediul experientelor, sa se auto organizeze atunci cand se confrunta cu o anumita problema si sa reflecte asupra momentelor de succes dar si asupra celor mai putin bune in vederea unei dezvoltari continue.

In comparatie cu *Agile*, *SCRUM* este o metoda efectiva de lucru in timp ce *Agile* este o metoda de gandire. *SCRUM* aplica principiile *Agile* in comunicarea si lucrul de zi cu zi, bazandu-se pe o invatare si ajustare continua in functie de anumiti factori fluctuanti. Luand in considerare faptul ca nu toate cerintele sunt cunoscute inca de la inceputul unui proiect, este structurat in asa fel incat sa ajute echipele sa se adapteze in mod natural la schimbari de business si cerinte alte utilizatorilor, motiv pentru care parti functionale ale unui proiect sunt livrate dupa iteratii scurte de perioade predeterminate, existand posibilitatea reprioritizarii functionalitatilor pe parcurs.



SCRUM - Artefacte

artefáct2 *sn* [At: MDA / Pl: *~e* / E: eg ***artefact***] Obiect produs de activitatea umană.

- **Product Backlog** - Lista principala de responsabilitati care trebuie indeplinite de catre echipa, mentinuta de *Product Owner* sau *Product Manager;* Este o lista dinamica care include cerinte, imbunatatiri, lucruri care trebuie reparate (*bugs, fixes*) si care serveste ca *input* pentru *Sprint Backlog*. Acest *backlog* este revizuit constant, reprioritizat si mentinut de catre *Product Owner* deoarece, pe masura ce se descopera lucruri noi care sa imbunatateasca produsul, ori piata se schimba, unele puncte din lista s-ar putea sa nu mai aiba valoare si anumite probleme sa fi fost rezolvate in alte modalitati.

SCRUM - Artefacte

artefáct2 *sn* [At: MDA / Pl: *~e* / E: eg ***artefact***] Obiect produs de activitatea umană.

- **Sprint Backlog** - Lista curenta de responsabilitati, *user stories*, *bug fixes,* selectate de catre echipa de dezvoltare pentru a fi implementate in iteratia curenta - **SPRINT**. Inainte de fiecare *sprint,* are loc o sedinta de *Sprint Planning* in cadrul careia echipa alege cantitatea de lucru posibila si necesara pentru *sprint*-ul curent. Acest *backlog* poate fi flexibil si poate fi modificat pe parcurs, insa fara a afecta ceea ce echipa decide ca fiind scopul sprint-ului (*Sprint Goal*)

SCRUM - Artefacte

- **Sprint Goal** - partea de functionalitate utilizabila care este livrata la finalul unui *sprint,* sau ceea ce echipa isi propune sa realizeze pana la finalul *sprint*-ului. Deobicei, aceasta este prezentata in cadrul unei sedinte ce poarta denumirea de **DEMO,** la finalul fiecarei iteratii, atunci cand echipa arata ce a reusit sa realizeze in perioada corespunzatoare. *Sprint Goal* este deasemenea stabilit in cadrul sedintei de *Sprint Planning,* iar de cele mai multe ori poate fi regasit alaturi de conceptul de **Definition of Done** care variaza de la echipa la echipa - acesta se refera la pasii si cerintele pe care trebuie sa le indeplineasca o anumita parte de functionalitate pentru a fi considerata *Done,* sau mai bine zis, gata pentru a fi folosita de catre utilizatori

SCRUM - Ceremonii, Evenimente

Cateva dintre cele mai cunoscute parti componente ale **SCRUM** sunt aceste evenimente secventiale, ceremonii sau sedinte pe care echipa le pune in practica in mod regulat. In cadrul acestora se evidentiaza cel mai bine variatiile de la echipa la echipa. Spre exemplu, unele echipe privesc toate aceste *meeting*-uri ca fiind greoaie, repetitive si consumatoare de timp, in timp ce altele le vad ca fiind absolut necesare.

- **Grooming / Organizare the backlog** - Eveniment responsabilitate a *Product Owner*-ului: principala lui preocupare este sa duca produsul in directia potrivita si sa aiba un impact constant atat asupra pietei cat si a utilizatorului, clientului. Astfel, el mentine *Product Backlog* atat bazat pe *feedback*-ul primit de la echipa de dezvoltare, cat si de la utilizatori, prioritizand si asigurand claritatea acestei liste astfel incat cerintele specificate sa fie pregatite pentru a fi implementate oricand

SCRUM - Ceremonii, Evenimente

- **Sprint Planning -** Cantitatea de lucru care trebuie acoperita pe pacursul *sprint*-ului curent este planificata in cadrul acestui *meeting* de catre intreaga echipa de dezvoltare. Sedinta este condusa de catre *Scrum Master* si acesta e momentul in care echipa decide *Sprint Goal*. Multiple *stories* care puse cap la cap duc la indeplinerea acestuia sunt adaugate in *sprint* din cadrul *Product Backlog*, iar echipa cade de comun acord asupra fezabilitatii acestora (pot / nu pot fi livrate). La finalul aceste sedinte, fiecare membru al echipei trebuie sa cunoasca in mod clar ce anume poate fi livrat si care este scopul care trebuie atins pentru a duce iteratia la bun sfarsit

SCRUM - Ceremonii, Evenimente

- **Daily Scrum / Stand-up** - este o sedinta scurta (~15 minute) care are loc zilnic in acelasi cadru si la aceesi ora (deobicei dimineata). Scopul acesteia este alinierea echipei cu privire la lucrurile care s-au intamplat in ultimele 24 de ore - impedimente, noutati, status - si planificarea pentru ziua in curs.

  DSU este momentul perfect pentru membrii echipei sa decida si sa exprime anumite incertitudinea cu privire la indeplinirea *Sprint Goal*. O metoda obisnuita de desfasurare a *stand-up*-ului este ca fiecare membru al echipei sa raspunda la 3 intrebari:

  - Ce am facut ieri ?
  - Ce planuiesc sa fac astazi ?
  - Exista impedimente ?

SCRUM - Ceremonii, Evenimente

- **Sprint review** - La finalul fiecarui *sprint*, echipa se aduna in vederea unei intalniri informale pentru a vizualiza un *DEMO* al partilor functionale dezvoltate pe parcursul iteratiei. Se prezinta sarcinile din *backlog* care acum au statusul *DONE stakeholder*-ilor si membrilor echipei pentru a oferi feedback. *PO*-ul poate decide daca functionalitatile dezvoltate pot fi deschise sau nu spre a fi folosite de catre utilizatori (*Release*).

  Acest meeting poate fi premergator unei revizuiri a *backlog*-ului de catre *PO*, bazat pe *sprint*-ul curent, iar ulterior are loc sesiunea de *Sprint Planning* pentru urmatoarea iteratie.

SCRUM - Ceremonii, Evenimente

- **Sprint retrospective** - Retrospectiva este sedinta in cadrul careia echipa documenteaza si discuta ce a mers bine si ce nu in cadrul *sprint*-ului, proiectului, referitor la comunicarea in echipa, uneltele folosite sau chiar si cu privire la anumite sedinte. Ideea in sine este crearea unui mediu in cadrul caruia echipa sa se poata focusa pe ce a functionat sau ce trebuie imbunatatit pentru urmatoarea iteratie si mai putin pe ce a fost gresit.

Referinta: https://www.atlassian.com/agile/scrum

Kanban

este un alt framework popular care se bazeaza pe principiile *Agile*. Se focuseaza in principal pe livrarea continua de functionalitati si ajuta echipele sa lucreze impreuna mult mai eficient, bazandu-se pe comunicarea in timp real a capacitatii disponibile si transparenta. Responsabilitatile si sarcinile care trebuie indeplinite sunt reprezentate vizual sub forma de carduri pe un *kanban board,* permitand fiecarui membru sa vada oricand care este statusul dezvoltarii anumitor parti componente: progres, cine e responsabil, detalii tehnice, timp necesar/ramas, etc....

Kanban = “visual sign”

Team Kanban Board

QUICK FILTERS: Critical partners | Only my partners | Recently updated

Release

1 To do	4 In progress	3 Code review Max 2	1 Done
TIS-28 Research options to travel to Pluto	TIS-25 Engage Jupiter Express for travel	TIS-27 Engage Saturn Resort as PTP	~~TIS-23~~ Engage JetShuttle SpaceWays for travel
	TIS-25 Add Deimos Tours as a travel partner	TIS-27 Engage Speedy SpaceCraft	
	TIS-20 Engage Saturn Lines for group tours	TIS-26 Reach out to the Red Titan Hotel	
	TIS-24 Sign Contract for SunSpot Tours		

Kanban - Avantaje

- **Planning flexibility** - echipa se focuseaza in principal pe lucrurile care sunt inca in progres. Atunci cand un task este indeplinit, urmatorul *task* in ordinea prioritatii din cadrul 'TO-DO' este pus in progres. *PO*-ul poate sa reprioritizeze orice lucru din *to-do-list*, fara a distrage echipa de la lucrurile care sunt in progres. Atat timp cat lucrurile sunt prioritizate corect, echipa este asigurata de faptul ca ei livreaza lucrurile cele mai importante din punct de vedere al business-ului. Asadar, nu mai este nevoie de acele iteratii de durata fixa intalnite in *Scrum*.

Kanban - Avantaje

- **Shortened time cycles** - “Cycle time” reprezinta cantitatea de timp necesara pentru ca o unitate de lucru sa treaca prin *workflow*-ul echipei, din acel moment in care lucrul este in progres pana in momentul in care functionalitatea se livreaza. Prin optimizarea acestui proces, echipa poate prezice cu usurinta timpul necesar pentru livrarea altor viitoare functionalitati.
  Deoarece fiecare membru al echipei are un set de capabilitati diverse si nu se concentreaza doar pe o anumita arie, oricine poate lucra la orice, iar astfel aceste cicluri au o durata de timp redusa pentru ca impedimentele sunt deblocate foarte usor. Spre exemplu, testarea nu este facuta doar de catre echipa de QA, *developer*-ii ajuta si ei la acest lucru. Este responsabilitatea intregii echipe sa se asigure ca flow-ul de dezvoltare decurge lin pe parcursul intregului proces.

Kanban - Avantaje

- **Fewer bottlenecks** - “Multitasking kills efficiency” - Cu cat mai multe lucruri sunt in progres, cu atat apar mai multe schimbari de context care se intercaleaza si amana completarea acestora. De aceea, un principiu de baza al *Kanban* este limitarea numarului de sarcini care sunt in progres. Astfel, impedimentele sunt foarte clare, iar procesul nu este afectat major de lipsa de experienta, capabilitati tehnice, lipsa de oameni, etc...

- **Continuous Delivery** - Livrarea de functionalitati in mod continuu, pe masura ce acestea sunt terminate

Referinta: https://www.atlassian.com/agile/kanban

Scrum vs Kanban

	SCRUM	KANBAN
Cadence	Regular fixed length sprints (ie, 2 weeks)	Continuous flow
Release methodology	At the end of each sprint if approved by the product owner	Continuous delivery or at the team's discretion
Roles	Product owner, scrum master, development team	No existing roles. Some teams enlist the help of an agile coach.
Key metrics	Velocity	Cycle time
Change philosophy	Teams should strive to not make changes to the sprint forecast during the sprint. Doing so compromises learnings around estimation.	Change can happen at any time

Scrum, Kanban, Agile

- *Scrum*, *Kanban* sunt framework-uri care implementeaza *Agile mindset*
- Exista si modele hibride adoptate de unele companii, precum *Scrumban* sau *Kanplan*, care este un *Kanban* bazat pe un *backlog*
- Ambele *framework*-uri folosesc metode vizuale (*boards*) pentru a urmari progresul; Ambele pun pret pe eficienta si impartirea *task*-urilor complexe in parti mai mici, usor de “manevrat”, insa felul in care decurge procesul este diferit
- *Scrum* se focuseaza pe iteratii de perioade predefinite, fixe; *Kanban* ajusteaza iteratiile din mers si sunt fixe doar initial
- *Kanban* nu este la fel de structurat in comparatie cu *Scrum* si este mult mai deschis la interpretare
- *Scrum* in sine este destul de simplu iar conceptele implicate sunt destul de usor de invatat si inteles; Rolurile sunt clare, iteratiile scurte mentin echipa motivata si utilizatorii fericiti ca pot vedea progres rapid