

# Introducere



- Curs
- Elemente privind dezvoltarea aplicatiilor software in telecomunicatii

# Software Engineering (Ingineria Programarii- IP) in contextul dezvoltarii aplicatiilor software

- Dezvoltarea de software avansat implica:
  - 1. Metodologii clasice ale IP
  - 2. Abordari bazate pe unelte de dezvoltare automata, semi-automata de software (methodologii formale – tehnologia *Blockchain* pentru *bitcoin*, *Ethereum Virtual Machine*, *K framework*)
  - 3. Concepte moderne ce provin din viata de zi cu zi integrate in dezvoltarea de software (*Organic computing*, *Swarm applications*, *choreography* – the logic of the message-based interactions among the participants are specified from a global perspective "Dancers dance following a global scenario without a single point of control", *orchestration* – the logic is specified from the local point of view of one single participant, called the *orchestrator* (*BPEL- Business Process Execution Language*), etc.)

- -de la marile companii la toate companiile
- -70-80% din pretul aplicatiilor complexe este softul, sau mai mult
- -Definitie IP:
- *Aspectele critice, unele chiar contradictorii legate de elaborarea programelor au determinat tendinte de abordare sistematica integralista a acestei probleme, conducand la formarea de concepte, principii, metode, instrumente de realizare a programelor, toate reunite sub denumirea de **ingineria programarii**.*
- -In accord cu **IEEE**, software engineering (IP) e definita ca:  
*"the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software."*
- <http://ecomputernotes.com/software-engineering/what-is-software-engineering-write-basic-objective-and-need-for-software-engineering>
- - **Strategia** si **lucrul in echipa** e de luat in considerare

# Strategia



- Strategia urmareste trei etape principale :
- 1. stabilirea fazelor si etapelor prin care trece succesiv un program in timpul ciclului sau de viata precum si definirea continutului concret al acestor faze si etape care sa fie clar demarcate
- 2. elaborarea/utilizarea unor tehnologii, adica a unor metode si instrumente care sunt asociate etapelor
- 3. elaborarea unor metode stiintifice adica a unor metode adecvate pentru organizarea si conducerea diferitelor activitati legate de realizarea in stil industrial a aplicatiilor soft



□ **IP se refera la** utilizarea:

□ - in mod disciplinat,

□ - sistematic, si

□ - cu pricepere, a unor metode si  
instrumente software adecvate,

avand in vedere Obiectivele si Principiile de  
baza.

# Obiectivele generale ale IP



- A) Adaptabilitatea programelor
- B) Eficienta programelor
- C) Fiabilitatea programelor
- D) Perceptibilitatea programelor

# Principiile generale ale IP

- 1. modularitate
- 2. abstractizare
- 3. localizare
- 4. incapsulare
- 5. uniformitatii
- 6. completitudinii
- 7. confirmabilitatii
- ***Exemplu: Radical numar subunitar (laborator)***


# Metodologii IP

- Metodologiile de baza din IP folosite in dezvoltarea aplicatiilor dedicate sunt:
- - metodologii structurate bazate pe aplicatii procedurale,
- - metodologii formale, bazate pe matematica discreta
- - metodologii orientate obiect, bazate pe clase, polimorfism, mostenire, ...
- - metodologii bazate pe componente, servicii, micro-servicii, etc.
- - metodologii bazate pe framework-uri pentru RAD (Rapid Application Development) – folosite in general la dezvoltarea de software comercial
- - etc.



- Tendintele noi in dezvoltarea de software avansat introduc ideea de dezvoltare de **software in mod organic** (organic development).
- Acest lucru provine de la metafora cresterii organismelor vii, ce se face in mod natural.
- In domeniul calculatoarelor dezvoltarea organica de software implica utilizarea de unelte in realizarea de software, unelte ce sunt parte a aplicatiei insasi.
- Utilizatorul poate adauga/modifica trasaturi aplicatiei care nu au fost initial considerate de dezvoltator.
- Aceasta conditie este apropiata de cea legata de cresterea organismelor vii, motiv pentru care s-a introdus aceasta notiune astfel.
- **Swarm Intelligence** – SI. Aceste sisteme constau dintr-o populatie de agenti simpli care interactioneaza local intre ei si cu mediul. (Model albine)
- **Swarm communication**, este privit ca o extensie a OOP pentru comunicatii mobile în Internetul viitor
- (*The Circle* (2017 film) - Emma Watson, Tom Hanks)

# **Ciclul de viata al programelor (faze de baza)**



- Aplicatiile trebuie sa treaca prin urmatoarele faze de baza:
- -proiectarea
- -realizarea
- -testare
- -punere in functiune
- -exploatare
- -intretinere

- Fazele prin care trece o aplicatie pot fi considerate si astfel:
- -analiza cerintelor
- -elaborarea sarcinilor
- -proiectarea
- -implementarea/testarea
- -instalarea si verificarea
- -intretinerea

# Clasificarea programelor

- -simple
- -medii
- -complexe
- -deosebit de complexe
- -aproape imposibile

Productivitatea:

- val. Prod. in ore/chelt. Munca in ore
- val. Prod. (\$, Eur)/timp. Fabricatie (h, zile)

# Gestiunea (Managementul) proiectelor software



- Managementul proiectelor software în vedere:
  - - definirea etapelor uzuale ale unui proiect software
  - - înțelegerea nevoii de planificare, monitorizare și control
  - - măsurarea succesului unui proiect pentru îndeplinirea obiectivelor sale.
- Managementul proiectelor software include cunoștințele tehnice și echipamentele necesare pentru dezvoltarea și gestionarea unui produs software.

- Managementul proiectelor - rol extrem de important în cadrul unei companii; de el depinde succesul/eșecul proiectelor.
- Manager de proiect - cunoștințe din domeniul tehnic și din domeniul economic.
- Vom aborda și tehnologiile moderne folosite pentru managementul proiectelor:
  - optimizarea planificării și implementării proiectelor
  - rezultatele dorite, costuri minime, timp de implementare cât mai scurt.



□ Managementul implica activitati precum:

Planificare - ceea ce trebuie realizat;

Organizare – organizarea activitatii;

Incadrarea personalului – selectare oameni potriviti;

Conducere, indrumare – introducerea unor instructiuni de urmat;

Monitorizare – verificare progres;

Control – actiuni ce remedieaza problemele care apar;

Inovare – propunere noi solutii;

Prezentare – legatura cu utilizatorul;

# **Cerintele privitoare la managementul proiectelor soft**



- cerinte functionale
- cerinte de calitate
- cerinte referitoare la resurse



# Componente ale managementului proiectelor



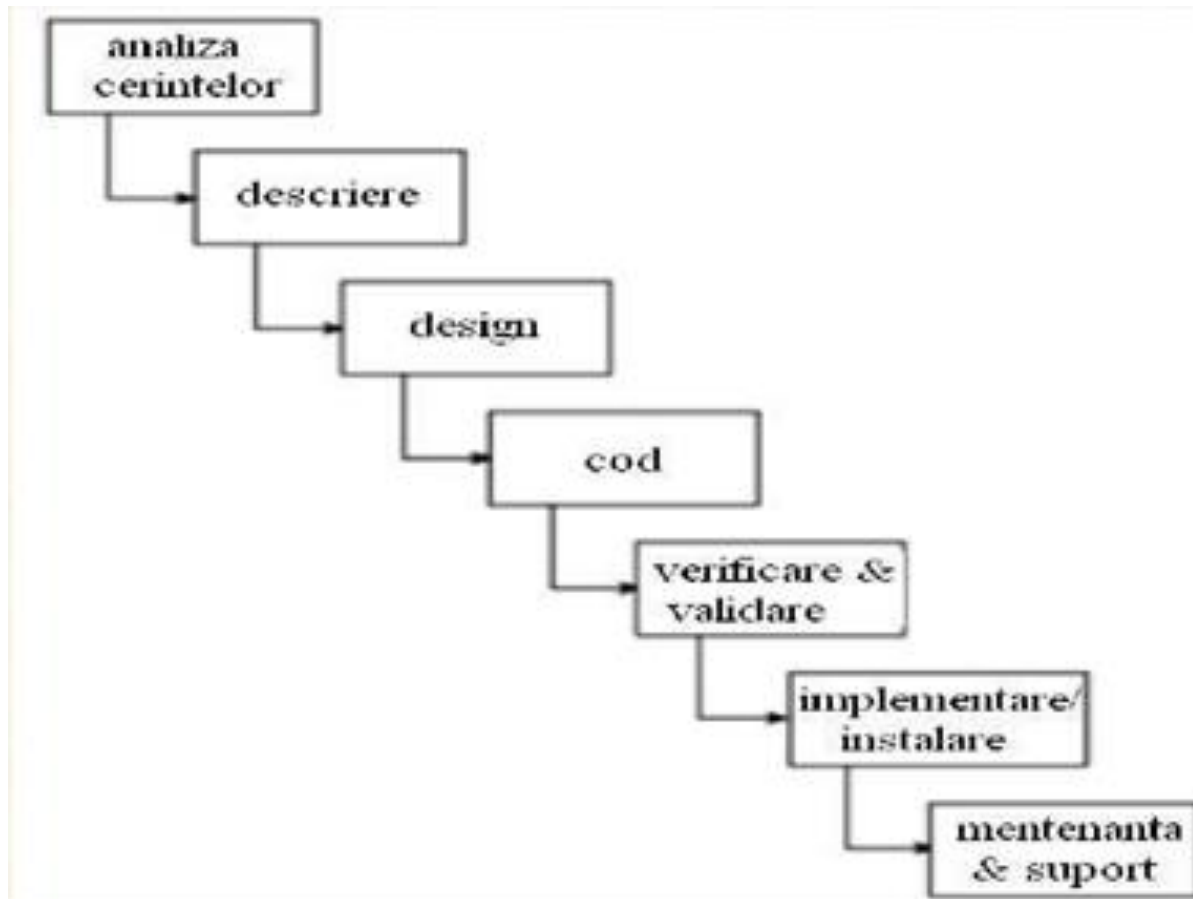
- A. Managementul sarcinilor
- B. Managementul echipei de lucru
- C. Managementul contextului de implementare
- D. Managementul mentenantei produsului

# **Activitatile de baza implicate in managementul produselor soft**




- A. Studiul de fezabilitate
- B. Planificarea
- C. Executarea proiectului

# Executarea secventiala a proiectului



# Calitatea produselor software

- Calitatea are in vedere toate etapele planificarii si executiei produselor soft. Pentru orice sistem software trebuie sa existe trei specificatii si anume:
  1. Specificatie functionala ce descrie ceea ce trebuie sa faca sistemul (serviciul oferit).
  2. Specificatie de calitate ce descrie cat de bine opereaza componentele.
  3. Specificatie de resurse care specifica suma cheltuita pentru sistem.



□ In acest domeniu se incearca sa se identifice calitati specifice produselor si calitatile software sunt grupate in trei multimi:

1. Calitati operationale - product operation qualities;
2. Calitati revizuibile - product revision qualities;
3. Calitati de tranzitie - product transition qualities;

- 1. Factorii ***product operation*** quality sunt:
  - **Corectitudinea:** marja pana la care programul satisface specificatiile si indeplineste obiectivele utilizatorului.
  - **Fiabilitatea:** marja pana la care se asteapta ca programul sa lucreze cu parametrii specificati la precizia dorita.
  - **Eficienta:** cantitatea de resurse ceruta de software.
  - **Integritatea:** marja pana la care accesul la software sau date al persoanelor neautorizate poate fi controlat. (privacy- GDPR norme europene valabile din mai 2018)
  - **Modul de utilizare:** efortul necesar pentru a invata, opera, pregati intrarea si interpreta iesirea.

□ 2. Factorii ***product revision*** quality sunt:

- **Intretinerea:** efortul necesar pentru localizarea si repararea erorii intr-un program operational;
- **Testabilitatea:** efortul necesar pentru a testa un program pentru a ne asigura ca lucreaza in parametrii specificati;
- **Flexibilitatea:** efortul necesar pentru a modifica un program operational;

□ 3. Factorii ***product transition*** quality sunt:

- **Portabilitatea:** efortul necesar pentru transferul programului de pe o configuratie hardware si un mediu software pe alta/altul.
- **Refolosirea:** marja pana la care un program poate fi utilizat in alte aplicatii;
- **Interoperabilitatea:** efortul necesar pentru a cupla un sistem cu altul;

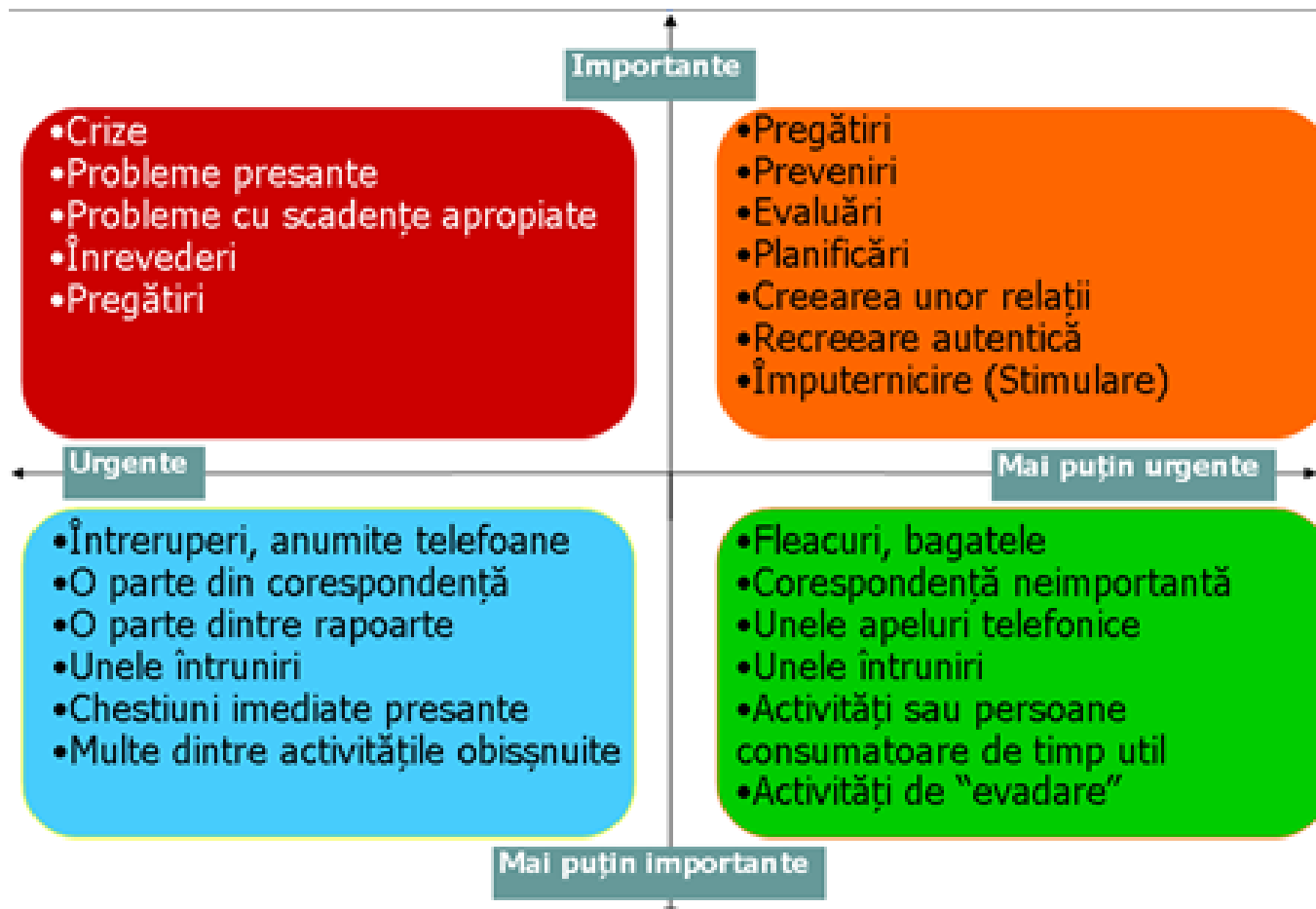


# Time management



- Pentru a ne organiza in mod eficient timpul este necesar sa urmam trei pasi:
1. crearea listei de prioritati;
  2. identificarea si marcarea importantei si urgentei fiecarei activitati din lista;
  3. stabilirea executiei actiunilor in timp;

# Patru sectoare de activitati avand in vedere urgenta si importanta evenimentelor



# **C2-Factorii umani si implicatiile in programare**



- IS – Information System
- IT – Information Technology
- ITC – IT + Communication

□ Revolutia IT, “electronic-communication cottages” – Alvin Toffler, Socul viitorului, Al treilea val (rev. agrara, industrială, stiintifica-tehnologica)

# Efecte IT – Legea cauza-efect

	Efect	Cauza
Individ	Demiterea expertului Somaj/reducerea timpului Munca dezumanizata Imbunatatirea muncii Imputernicirea raspunderii muncii	Sistem expert Automatizarea  Eliminarea calificarii Suportul dezvoltarii IT/IS
Societati	Colapsul unui oras Posibilitati de timp liber ridicat	Caderea comunicatiilor Automatizarea
Organizatii	Intreprinderi fara oameni Oficii fara hartii	Automatizarea  Rețele, baze de date

# Criterii ce duc la dezvoltarea IS

## □ A) Succes

- -productivitatea
- -eficienta

## □ B) Pene

- -tehnice
- -de utilitate – utility
- -de intrebuintare – usability

Eficacitatea, satisfactia, eficienta – triumphi

Factorii umani legati de alte stiinte: fiziologia, psihologia, sociologia, filozofia, spiritualitatea

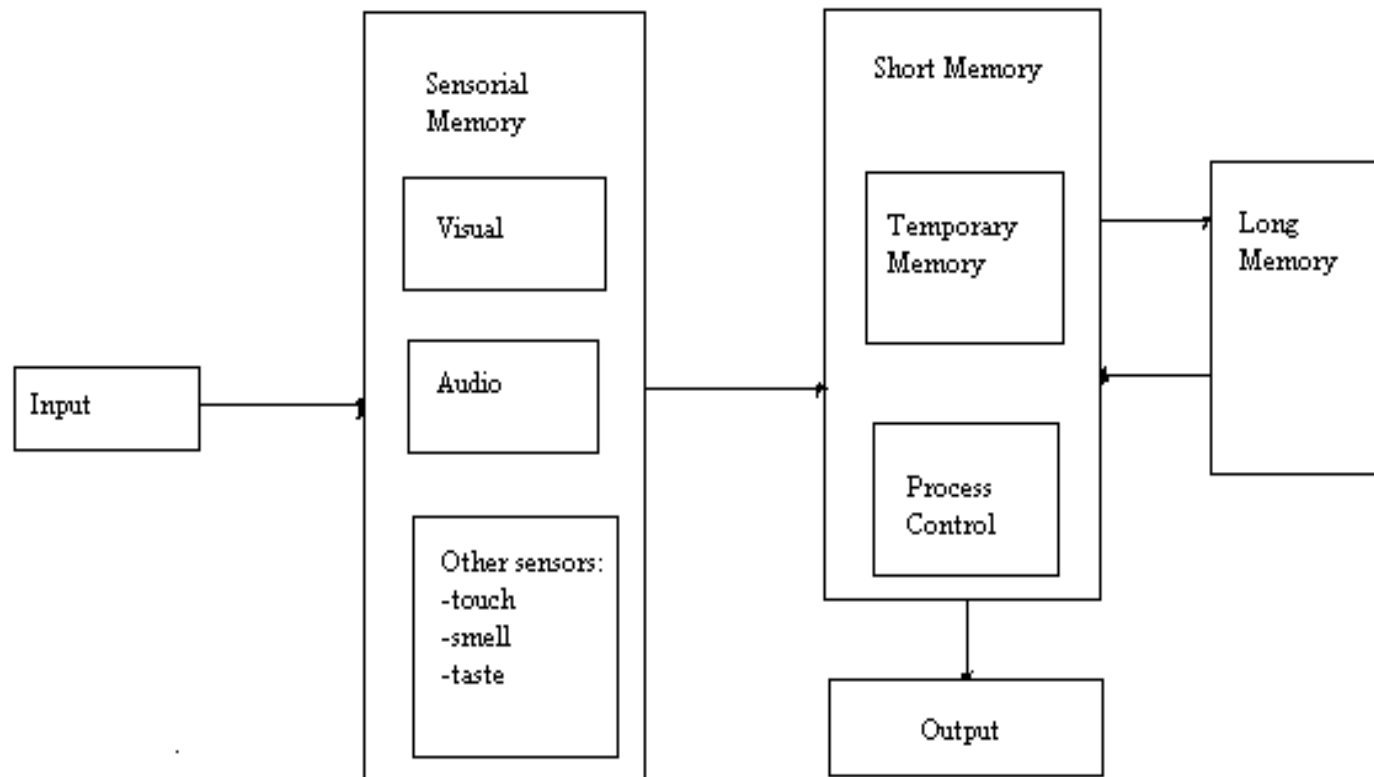
# **Psihologia si memoria umana**

## **□ Memoria umana:**

- -senzoriala
- -de scurta durata
- -de lunga durata

## **□ Procesoare la nivel uman:**

- -perceptual - simturi
- -cognitive -minte
- -motor - executie
- -spiritual – sinteza, intelepciune



# Nevoile utilizatorilor



- fizice
- functionale
- aspirationale



# **Cai de studiu in domeniul uman pentru dezvoltarea aplicatiilor software**



- 1. Observarea
- 2. Interviu
- 3. Colectare informatii etapa 1 si 2
- 4. Intrebari statistic relevante usor de completat
- 5. Metode participative
- 6. Experimente – story boarding based scenario

# Metodologia proiectarii IS

- 1. Identificare obiective
- 2. Identificare constrangeri;
  - 1, 2: Analiza
- 3. Identificare solutii alternative
- 4. Evaluare solutii alternative
- 5. Selectarea celei mai bune alternative
  - 3, 4, 5: Proiectare
- 6. Implementarea solutiei alternative alese

# Metodologii de clasificare

## □ A. Software

- -structurate
- -OO
- -formale
- -bazate pe componente, servicii si micro-servicii
- -metodologii bazate pe framework-uri pentru RAD (Rapid Application Development)
- -metode avansate de tip organic, swarm, etc.

## □ B. Sistemelor informatice

- -conducerea proceselor tehnologice
- -conducerea proceselor socio-economice
- -cercetare stiintifica si proiectare tehnologica, etc.

# Interfatarea si Implementarea.

## Principiul lui Parnas

- Sistemele soft vazute dpdv:
  - -al interfetei, fata vazuta de alti programatori si utilizatori
  - -modului de implementare, fata vazuta de programatorul care lucreaza efectiv
- **Separarea** intre *modul de interfatare* si cel de *implementare* reprezinta cel mai important concept al IP obiectuale
- Cantitatea de informatii schimbata intre programatori si sistem e de asemenea importanta

# **Reguli- principiul lui Parnas- dezvoltare/reutilizare componente**



- A. sa ofere toate informatiile dorite pentru a face utile serviciile si nimic in plus
- B. sa ofere toate informatiile necesare pentru a-si indeplini responsabilitatile asignate si nimic in plus

# Transformarea descrierii unei componente in implementare

- 1. proiectarea structurilor de date
- 2. transformarea comportamentului in algoritm
- 3. implementarea componentei – facilitator (lucreaza in spate)
- 4. integrarea componentelor de tip stub-uri acolo unde e nevoie, comportament limitat
- 5. testarea unei componente individuale- unitate de testare
- 6. integrarea testarii- inlocuire stub-uri
- 7. testare regresiva - daca se fac modificari
- 8. mentenanta

# Managementul dezvoltarii aplicatiilor software

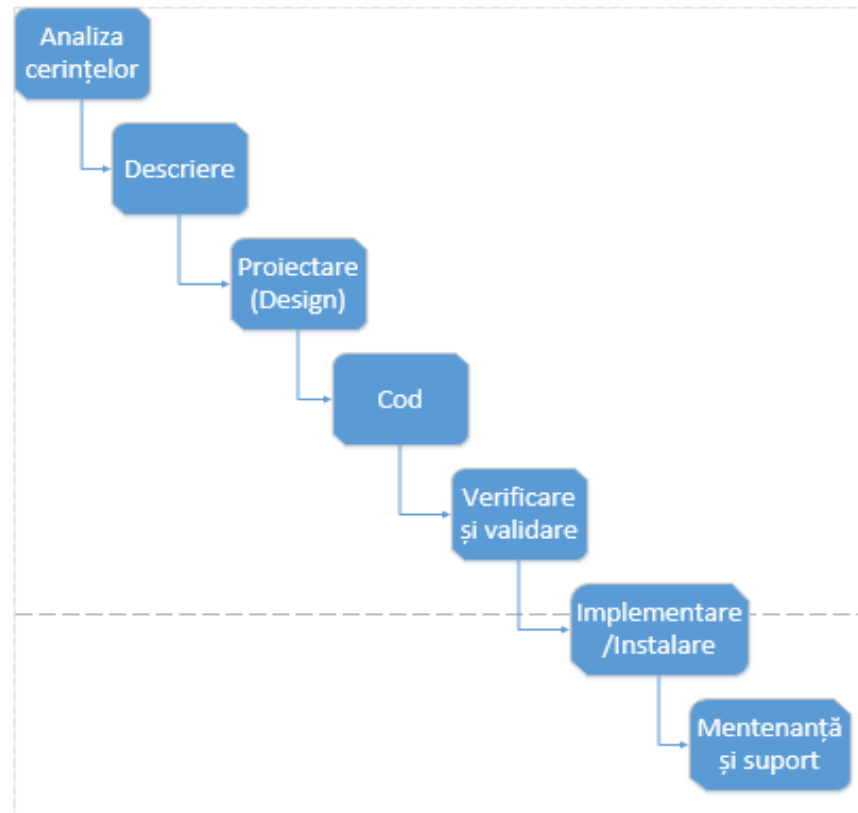
- Managementul sistemelor software împrumută masiv din managementul proiectelor domeniilor de baza, de unde a pornit, dar sunt nuante ce apar in domeniul software ce nu se gasesc in managementul altor discipline
- Procesul de dezvoltare software:
- Acest proces e intens dezbatut de realizatori. Cele mai cunoscute modele folosite in acest moment sunt:
- - modelul Waterfall (in casacada), modelul in V, ca si modele secventiale;
- - modelul Spiral (in spirala), modele de dezvoltare iterative si incrementale, etc.

# METODE DE DEZVOLTARE A PRODUSELOR SOFTWARE

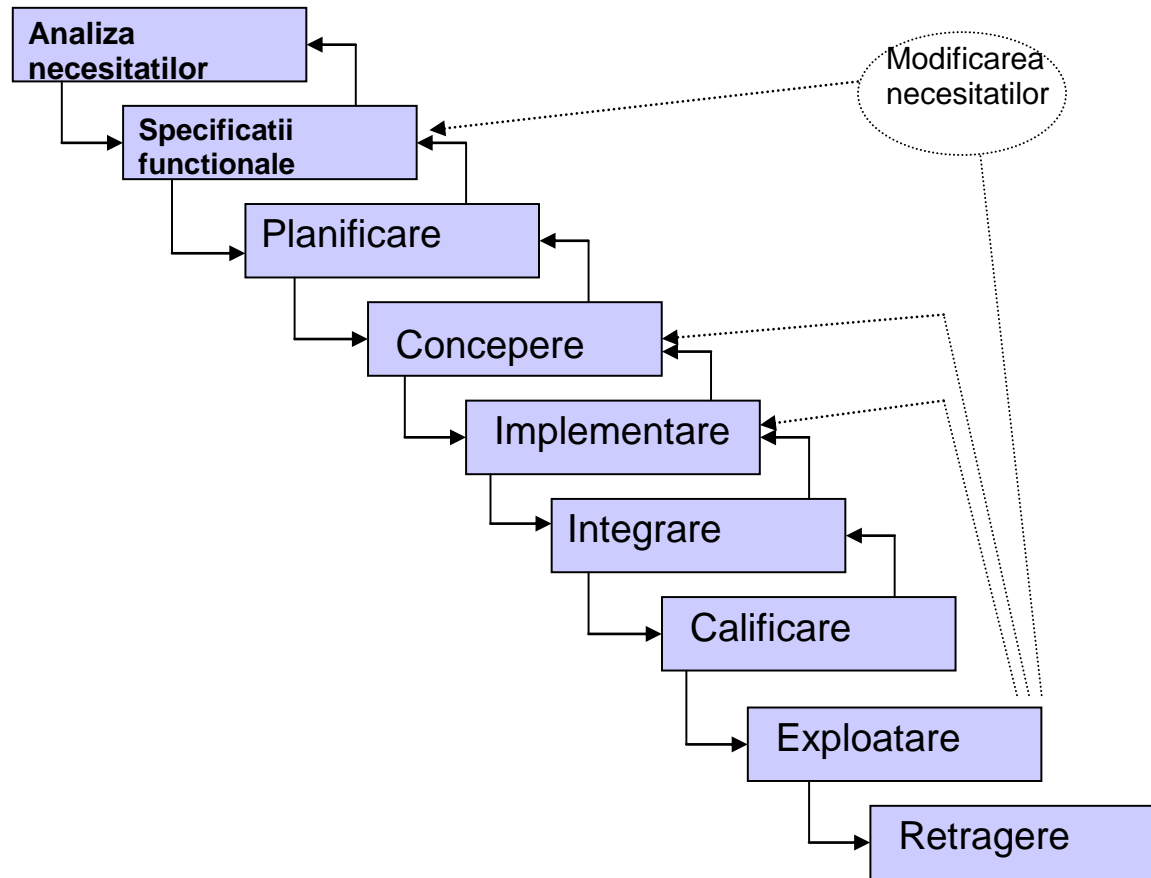
- Metode secvențiale - folosite în trecut (prototip, extensii succesive, în doi pași) – restrictive, predictive și cu preț fix;
- Acum sunt folosite ca și metode secvențiale:
  - Modelul în Cascadă (Waterfall)
  - Modelul în V
- Metodele spirale, iterative și incrementale - permissive și adaptive, permit schimbări în decursul proiectelor:
  - Modelul Agile/Scrum
  - Modelul Just – In – Time/Kanban

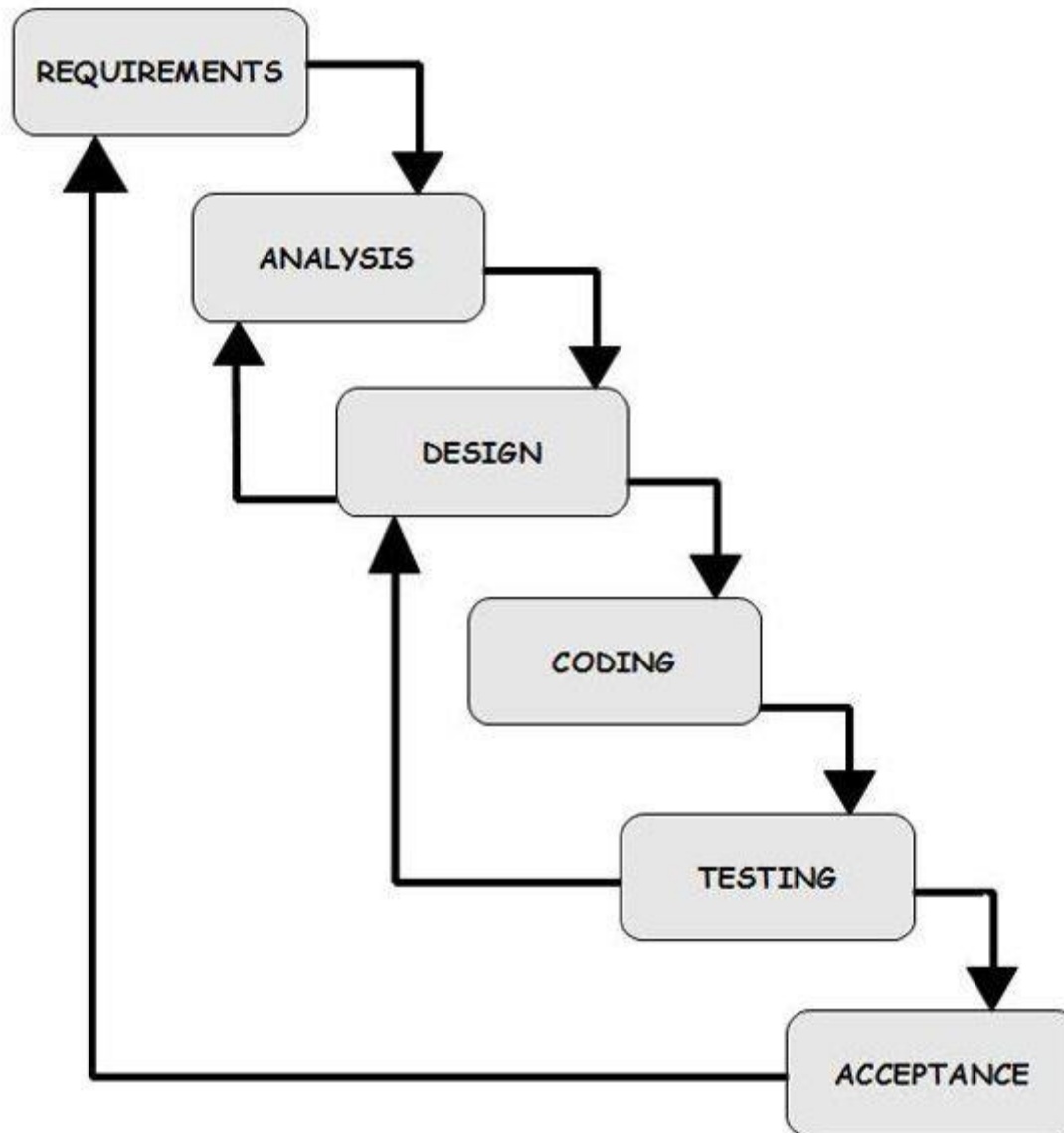


# Modelul în cascadă (waterfall) secvențial

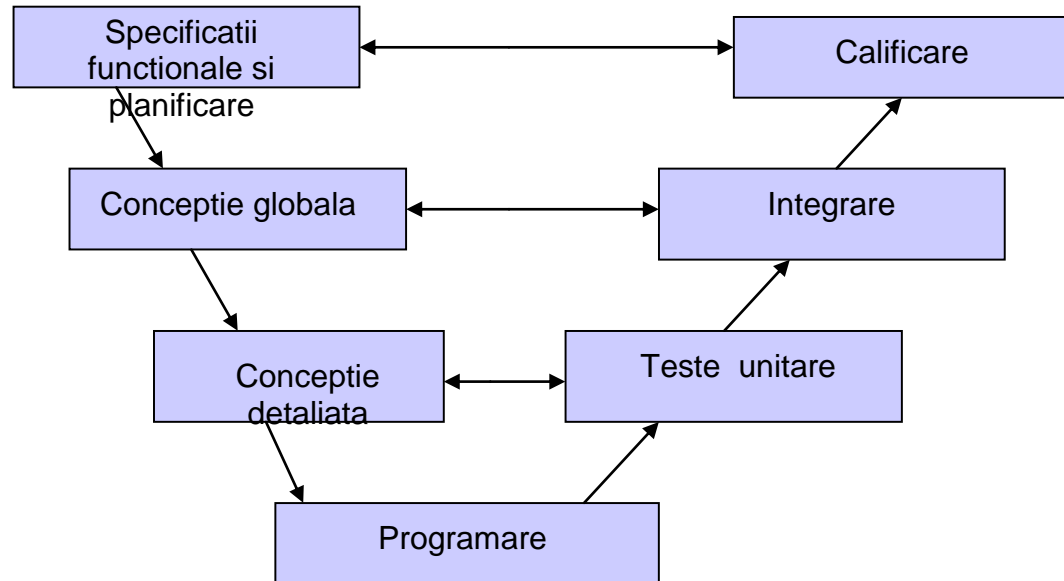


# Modelul in cascada cu feed-back

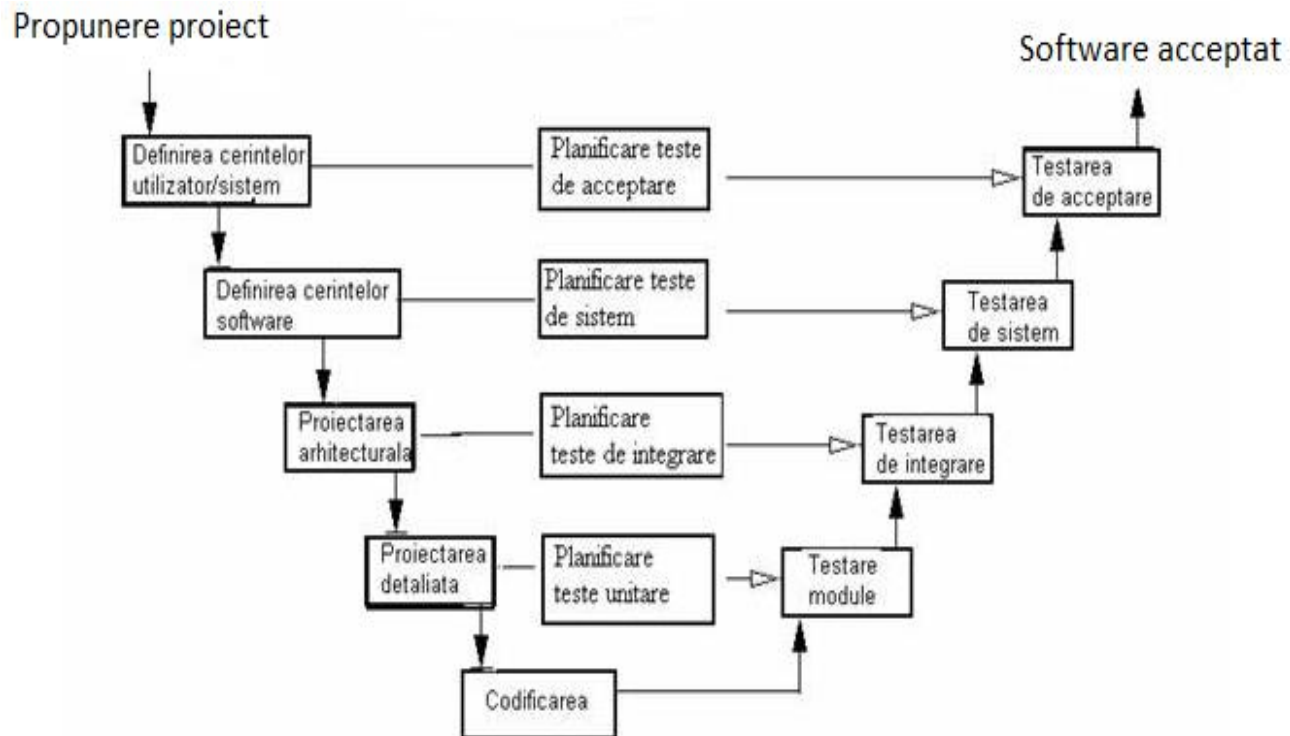




# Modelul in V



# Modelul în V

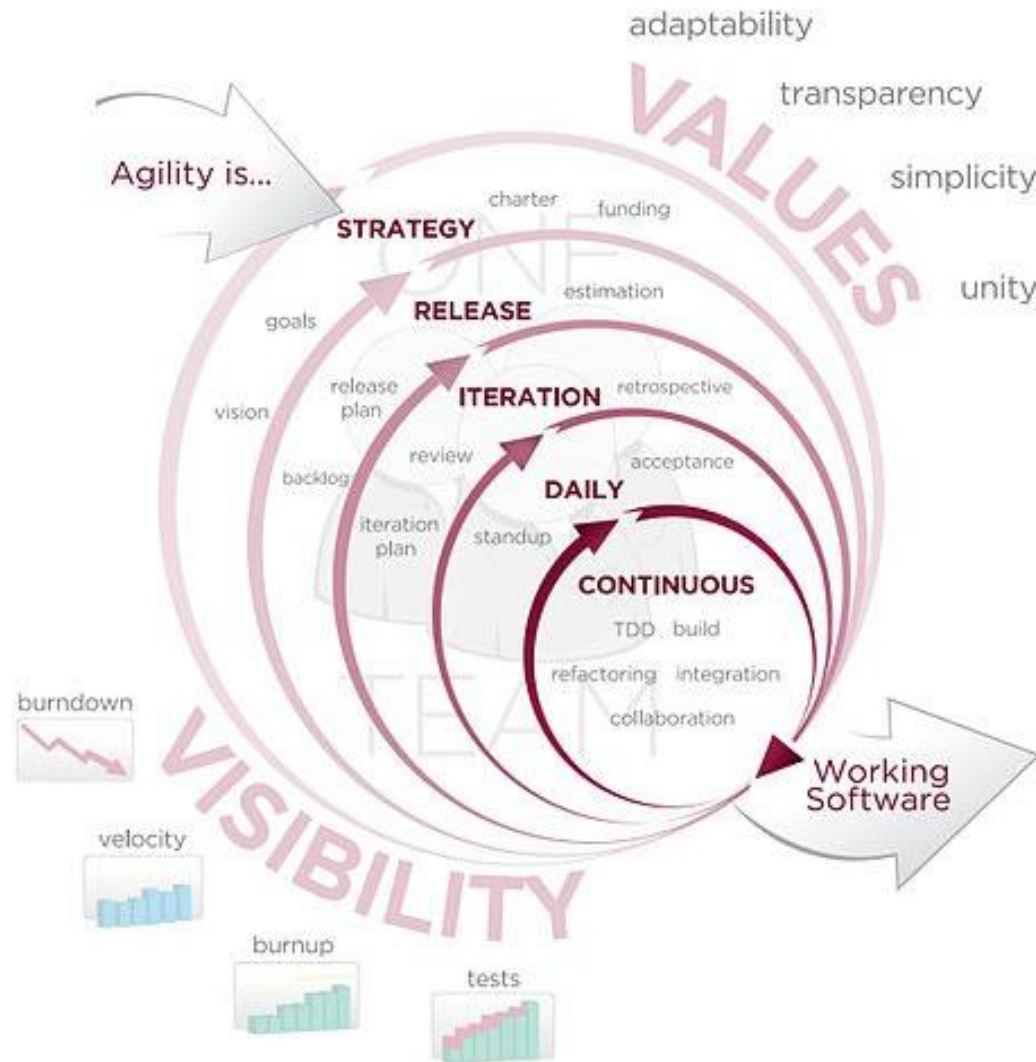


# Dezvoltarea de software Agile



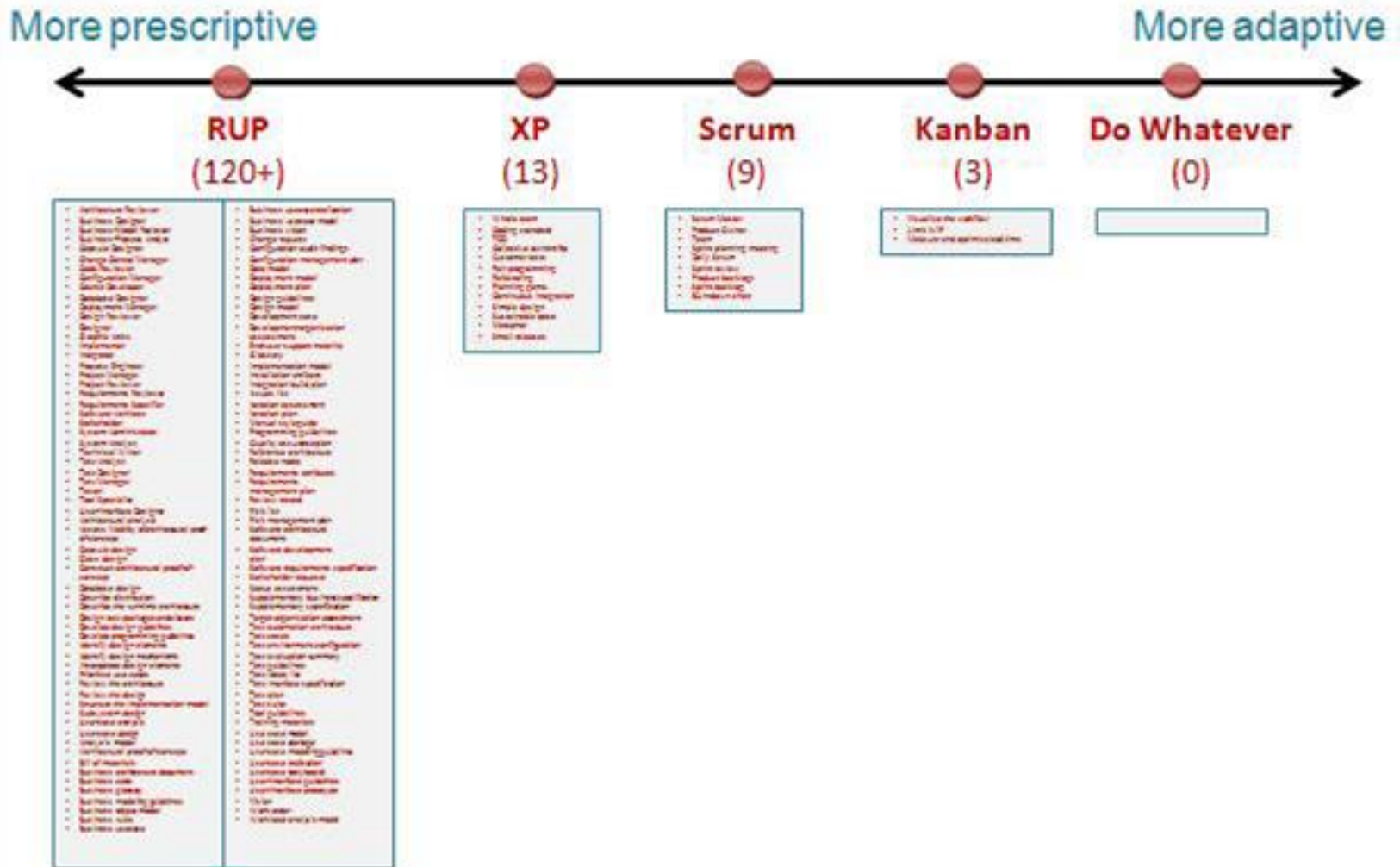
- **Dezvoltarea de software Agile** e realizata printr-un grup de dezvoltare de metodologii software bazat pe un proces de dezvoltare iterativ si incremental, unde cerintele si solutiile evolueaza catre colaborari intre echipe auto-organizate si cu cross-functionalitati (interdisciplinare).
- Exista un Agile Manifesto incepand cu anul 2001.
- Sunt 12 principii de baza considerate, de la satisfactia utilizatorului, la adaptari reglate functie de modificarile de circumstanta

# AGILE DEVELOPMENT



## ACCELERATE DELIVERY

# Metode diverse de la cele adaptive la cele prescriptive



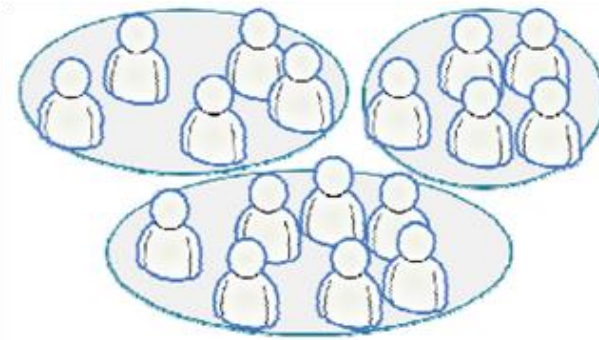


- Putem compara metodele pornind de la numarul de reguli care le furnizeaza. *Prescriptive* inseamana “mai multe reguli de urmat” si *adaptive* inseamna “mai putine reguli de urmat”.
- 100% prescriptiv inseamna ca nu iti mai folosesti creierul, ai o regula pentru orice.
- 100% adaptiv inseamna sa Faci Orice, nu sunt de loc reguli sau constrangeri. Amandoua extremele par a fi improprii din punct de vedere practic.
- Metodele Agile sunt uneori numite metode *lightweight*, in mod specific pentru ca sunt mai putin prescriptive decat metodele traditionale
- Scrum si Kanban sunt amandoua extrem de adaptive, dar vorbind in mod relativ, Scrum e mai prescriptiva decat Kanban.

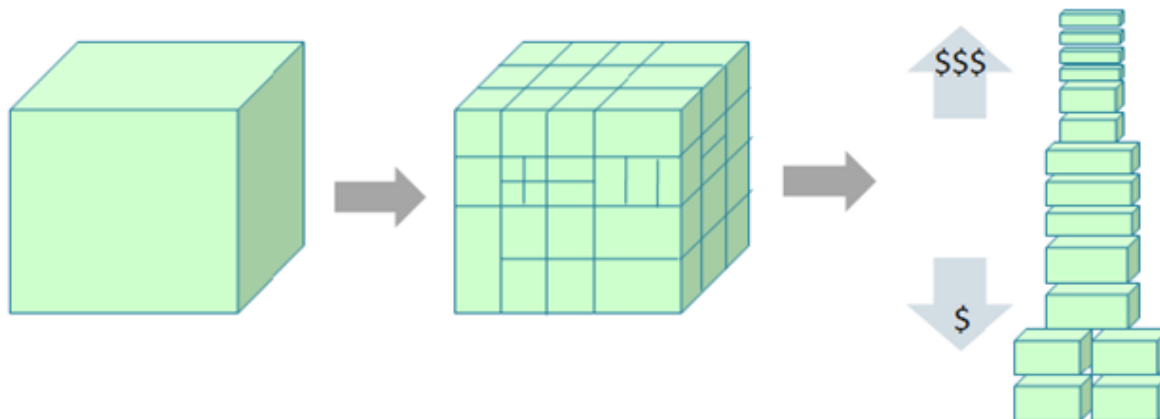
- **RUP** (Rational Unified Process) este cel mai prescriptiv – are peste 30 roluri, peste 20 activitati, si peste 70 artefacte. O cantitate imensa de lucruri de a învăța.
- **XP** (eXtreme Programming) este mai prescriptiv comparativ cu Scrum. Include multe din Scrum + un grup acceptabil de practici specifice ingineresti cum ar fi dezvoltarea bazata pe teste si programarea in pereche
- **Scrum** ofera mai multe constrangeri, si astfel lasa mai putine optiuni. Spre exemplu Scrum prescrie utilizarea iteratiilor de tip time-boxed pe cand, Kanban nu.
- **Kanban** lasa aproape totul deschis. Singurele constrangeri sunt Vizualizarea workflow-ului si limitarea munci in progres, WIP (Work In Progress). Destul de aproape de a Face Orice, dar surprinzator inca puternic.

# Metodologii de dezvoltare software - **SCRUM**

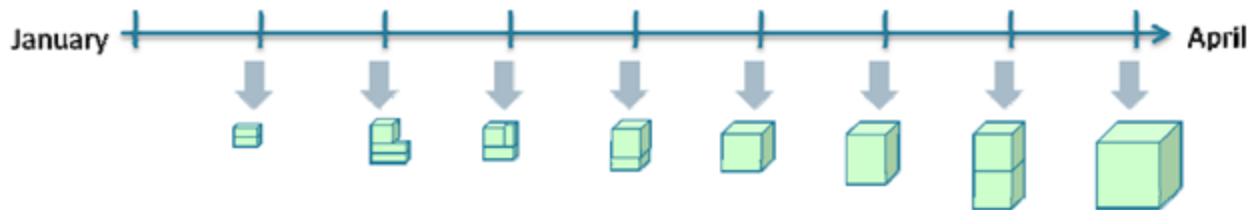
- **Scrum** vrea ca tu:
- Sa imparti organizatia in mici echipe interdisciplinare, cross-functionale, echipe ce se auto-organizeaza:



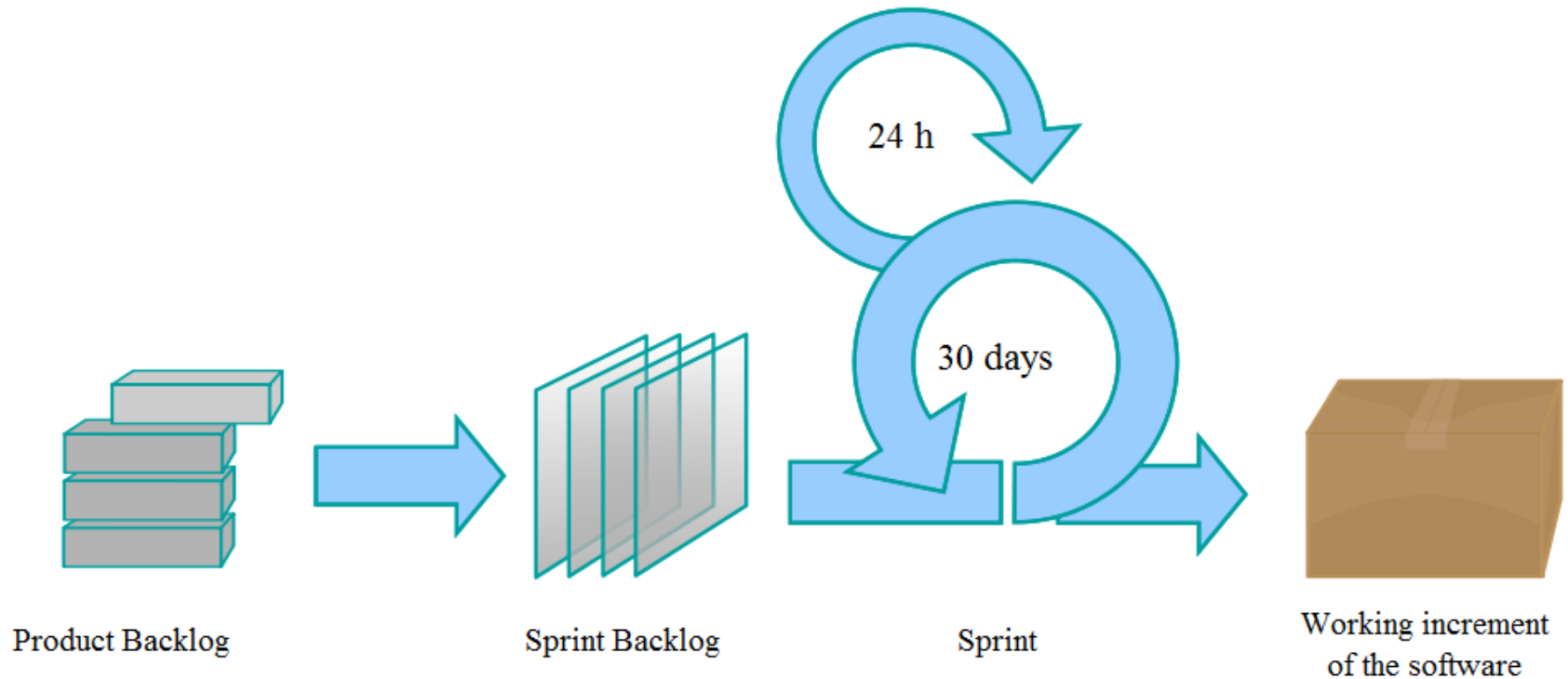
- Sa imparti munca intr-o lista de mici, elemente concrete



- Sa imparti timpul in iteratii scurte de lungime fixa (uzual 1 – 4 saptamani), cu potentialitate de a livra cod demonstrabil dupa fiecare iteratie.



# Concept Agile/Scrum

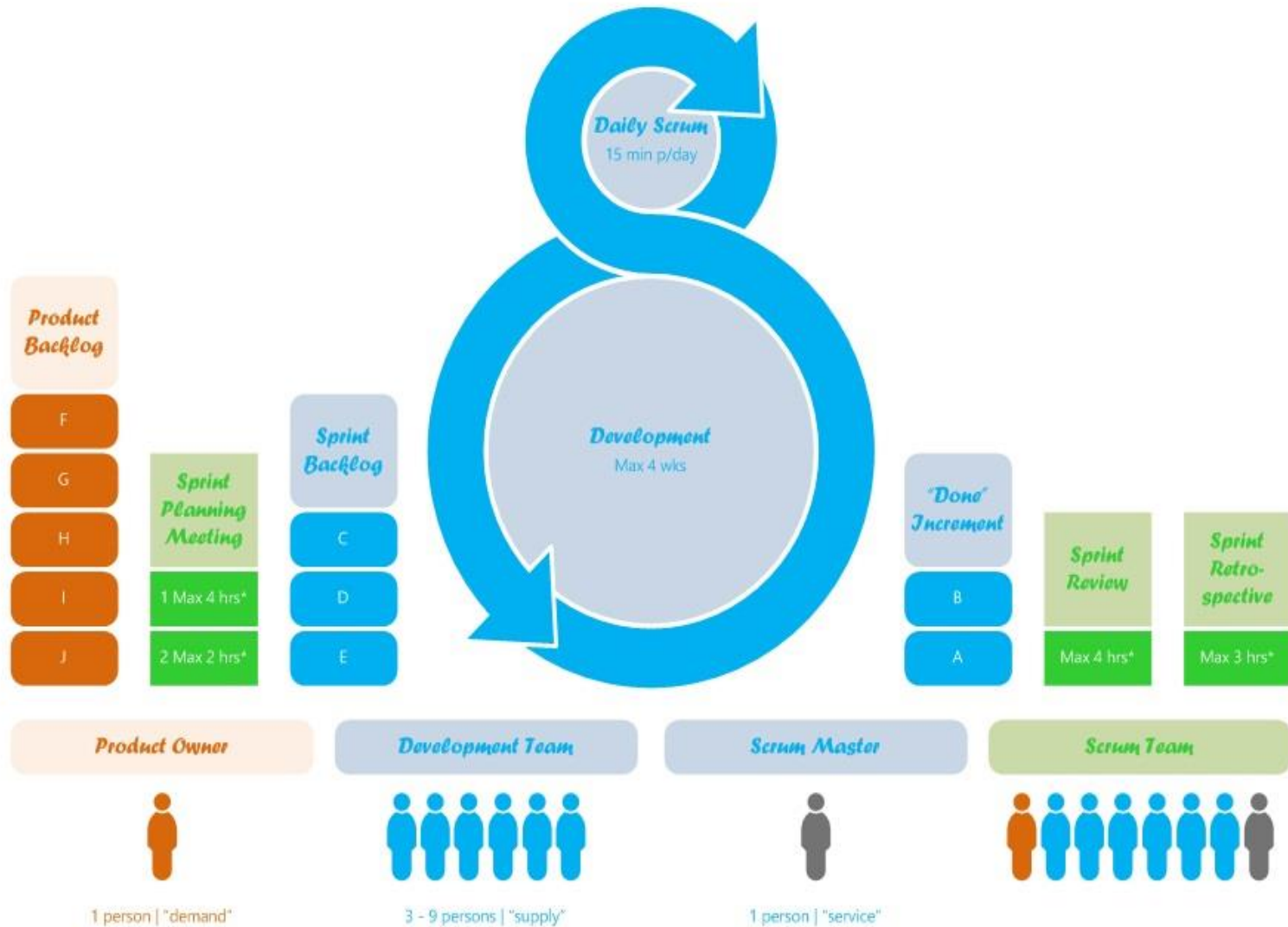


# Roluri Agile/Scrum

- PO – Product Owner – deținătorul produsului-legatura cu clientul intr-o abordare de afaceri.
- PM – Project Manager – cel ce coordoneaza proiectul (sau mai multe proiecte) cu abilitati tehnice si economice
- SM – Scrum Master – persoană care se asigură că procesul este respectat, că impedimentele sunt înlăturate precum și că fiecare membru al echipei are de lucru.
- Echipa Agile – echipa de dezvoltare – este responsabilă de proiectare și implementare (developer- tester).

- -abordarea clasica e in general de tip *tip waterfall* – (in cascada) folosind diagrame Gantt
- -la Agile/Scrum divizarea de tip SOW (Statement Of Work) : pachete mici
- *Echipele* de proiect (medii sau mici) :
  - **1 Product Director** rerprezentant Client, *Product Owner*
  - **1 Project Manager** (poate sa gestioneze unul sau mai multe proiecte deci nu neaparat membru) asigura legatura cu *SCRUM Master*
  - **2-6 dezvoltatori**, *echipa de dezvoltare*
  - **1 tester pentru fiecare 2 dezvoltatori**

# Scrum Overview

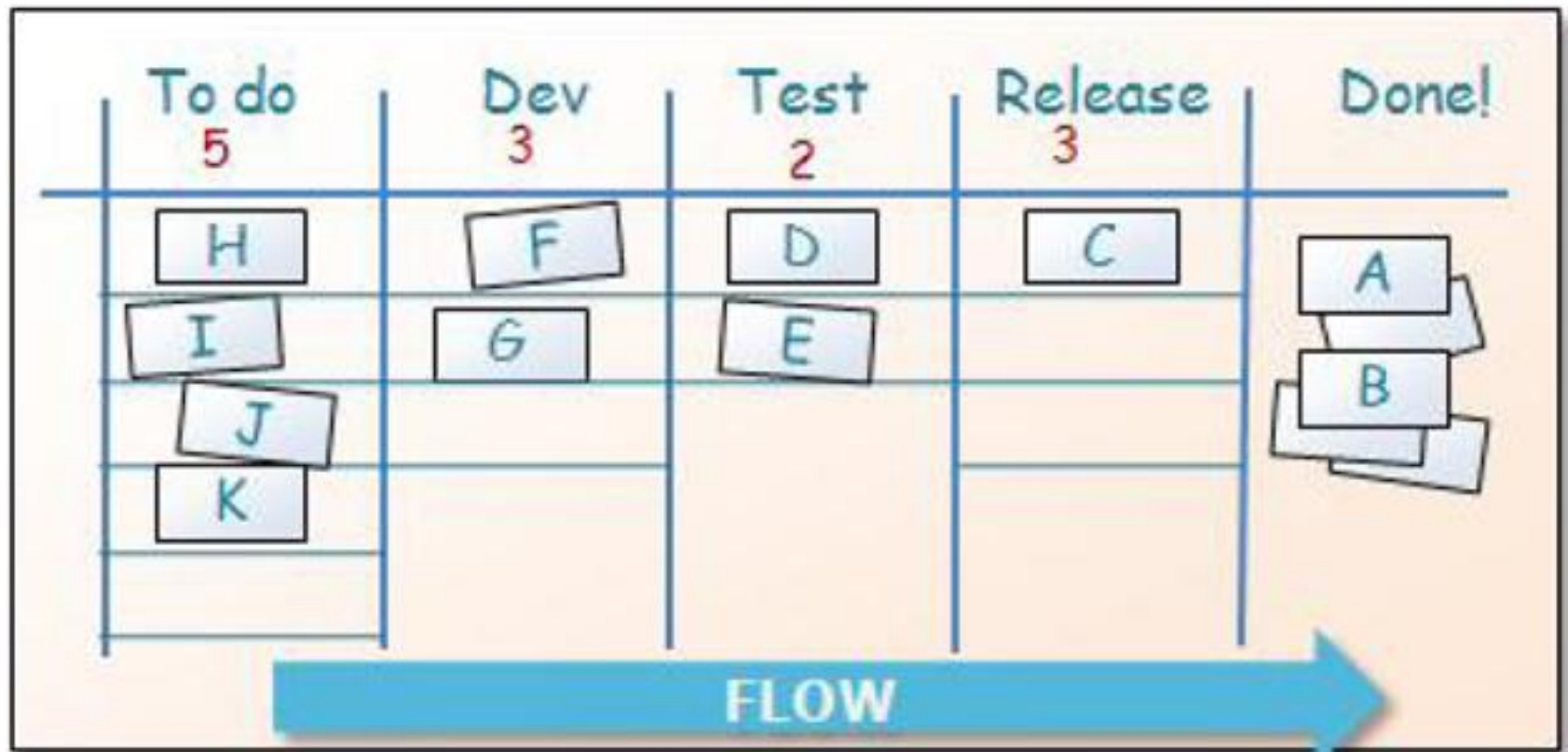


\* Duration of this event depends on the duration of the Sprint



# Kanban

- **Kanban vrea ca tu:**
- Sa vizualizezi fluxul de lucru (workflow-ul)
- Sa imparti munca in bucati, sa scrii fiecare actiune (item) pe un card si apoi sa pui pe perete (ex., state of art doctorat Franta, Calin Loghin)
- Sa folosesti coloane cu nume pentru a ilustra unde e fiecare item in workflow.
- Sa limitezi WIP (Work In Progress) – asignand explicit limite referitor la cate item-uri pot fi in progres la fiecare stare a workflow-ului.
- Sa masori asa numitul *lead time* (timpul mediu de realizare al unui item, uneori numit “cycle time”), sa optimizezi procesul pentru a face acest *lead time* cat de mic si predictibil posibl.



# Comparatie intre Agile/Scrum vs Kanban

## ➤ Asemănări:

- Amandoua sunt productive si de tip Agile/Just in Time
- Amandoua folosesc un mecanism de programare prin tragere, pull
- Amandoua limiteaza WIP (Work In Progress)
- Amandoua sunt transparente pentru a aduce imbunatatiri proceselor conduse
- Amandoua sunt orientate in a livra versiuni (release) software repede si des
- Amandoua sunt bazate pe propria organizare a echipelor
- Amandoua cer impartirea activitatii in bucati
- La amandoua, planul de *release* e continuu optimizat bazat pe date empirice (viteza / lead time)

# Comparație – Agile/Scrum vs. Kanban



## ➤ Asemănări:

- Ambele metode se bazează pe experiență (sunt empirice)
- Ambele metode permit lucrul pe mai multe produse simultan

# Comparație – Agile/Scrum vs. Kanban



## ➤ Deosebiri:

- Agile/Scrum este mai normativ decât Kanban
- Agile/Scrum impune iterații timebox
- Agile/Scrum impune existența rolurilor – PO, SM, Echipa Agile
- Limitarea WIP (Work In Progress)
- Agile/Scrum nu permite schimbarea într-o iterație
- Tabla Agile/Scrum se resetează între iterații
- Agile/Scrum impune echipe trans-disciplinare
- Sarcinile sunt împărțite în sprinturi pentru Agile/Scrum
- Agile/Scrum impune estimarea și viteza

# Diferente

Scrum	Kanban
Timeboxed iterations prescribed.	Timeboxed iterations optional. Can have separate cadences for planning, release, and process improvement. Can be event-driven instead of timeboxed.
Team commits to a specific amount of work for this iteration.	Commitment optional.
Uses Velocity as default metric for planning and process improvement.	Uses Lead time as default metric for planning and process improvement.
Cross-functional teams prescribed.	Cross-functional teams optional. Specialist teams allowed.
Items must be broken down so they can be completed within 1 sprint.	No particular item size is prescribed.
Burndown chart prescribed	No particular type of diagram is prescribed
WIP limited indirectly (per sprint)	WIP limited directly (per workflow state)
Estimation prescribed	Estimation optional
Cannot add items to ongoing iteration.	Can add new items whenever capacity is available
A sprint backlog is owned by one specific team	A kanban board may be shared by multiple teams or individuals
Prescribes 3 roles (PO/SM/Team)	Doesn't prescribe any roles
A Scrum board is reset between each sprint	A kanban board is persistent
Prescribes a prioritized product backlog	Prioritization is optional.

# Descriere produs JIRA

- **JIRA** – unealtă software folosită pentru planificarea și urmărirea unui proiect; este folosită atât pentru metodologia Agile/Scrum, cât și pentru Kanban.
- Alte unelte folosite pentru Agile/Scrum:
  - Agile Tracking Tool, Kunagi, Scrum Do, PangoScrum, Express, Scrummy, Sprintometer, Agilo for Scrum, etc.
- Alte unelte folosite pentru Kanban:
  - Kanbanize, Agile Zen, Kanbanery, Kanban Tool, Kanban Pad, etc.

# Descriere produs JIRA



- JIRA - trunchiere a cuvântului "Gojira", varianta japoneză a cuvântului Godzilla
- Dezvoltat începând cu anul 2002
- Cuprinde trei zone:
  - Zona de task-uri (sarcini)
  - Zona metricilor
  - Zona graficelor
- JIRA + plugin-ul GreenHopper – folosite pentru planificarea proiectelor.



# REZULTATE EXPERIMENTALE – Aplicatie mobila – Agile/Scrum

## Scrum board pentru primul sprint

The screenshot displays a JIRA Scrum board for the project 'Mobile Application - Scrum'. The board is organized into three columns: 'To Do', 'In Progress', and 'Done'. The 'To Do' column contains two tasks: 'SCRUM-6' (As a user, I'd like to have a shopping bag) and 'SCRUM-7' (As a manager for the site, I want to have more paying options for my customers). The 'In Progress' column contains two tasks: 'SCRUM-3' (As a user visiting the site, I'd like to be able to see the products present on the site) and 'SCRUM-4' (As a successful logged in user on the site, I'd like to be able to buy). The 'Done' column contains four tasks: 'SCRUM-2' (As a user, I'd like to be able to login), 'SCRUM-9' (As a team, I'd like to estimate the effort of a story in Story Points so we can understand the work remaining), 'SCRUM-14' (As a team, I'd like to commit to a set of stories to be completed in a sprint (or iteration)), and 'SCRUM-21' (Update task status by dragging and dropping from column to column >> Try dragging this task to "Done"). The board also includes a 'Quick Search' bar, a 'Create issue' button, and a 'Plan Work Report' section.

# REZULTATE EXPERIMENTALE

## – Agile/Scrum

### □ Story points pentru primul sprint

The screenshot displays the JIRA Agile interface for a project named 'Mobile Application - Scrum'. The top navigation bar includes tabs for 'Agile Project Management', 'Working with Epics', 'Story Point - GreenH', 'Mobile Application', and 'Creating a Board'. The main header shows the project name and a 'Create Issue' button. Below the header, there are tabs for 'Plan', 'Work', and 'Report'. The left sidebar shows a list of versions and epics. The main content area is divided into two sections: 'Sprint 1' and 'Backlog'. The 'Sprint 1' section shows a list of issues with their story points. The 'Backlog' section shows a list of issues with their story points. The right sidebar shows details for the selected issue, including its status, component, labels, and assignee.

Issue ID	Description	Story Points
SCRUM-50	As a team, I'd like to estimate the effort of a story in Story Points so we can understand the work remaining.	2
SCRUM-43	As an Agile team, I'd like to learn about Scrum.	2
SCRUM-44	As a user, I'd like to be able to create an account.	5
SCRUM-45	As a user, I'd like to be able to login.	4
SCRUM-46	As a user visiting the site, I'd like to be able to see the products present on the site.	1
SCRUM-47	As a successful logged in user on the site, I'd like to be able to buy.	1
SCRUM-48	As a user, I'd like to be able to pay for my products.	1
SCRUM-49	As a user, I'd like to have a shopping bag.	1

Issues: 8 Remaining: 2h 9m Estimate: 17

Backlog: 0

There are currently no issues in the backlog

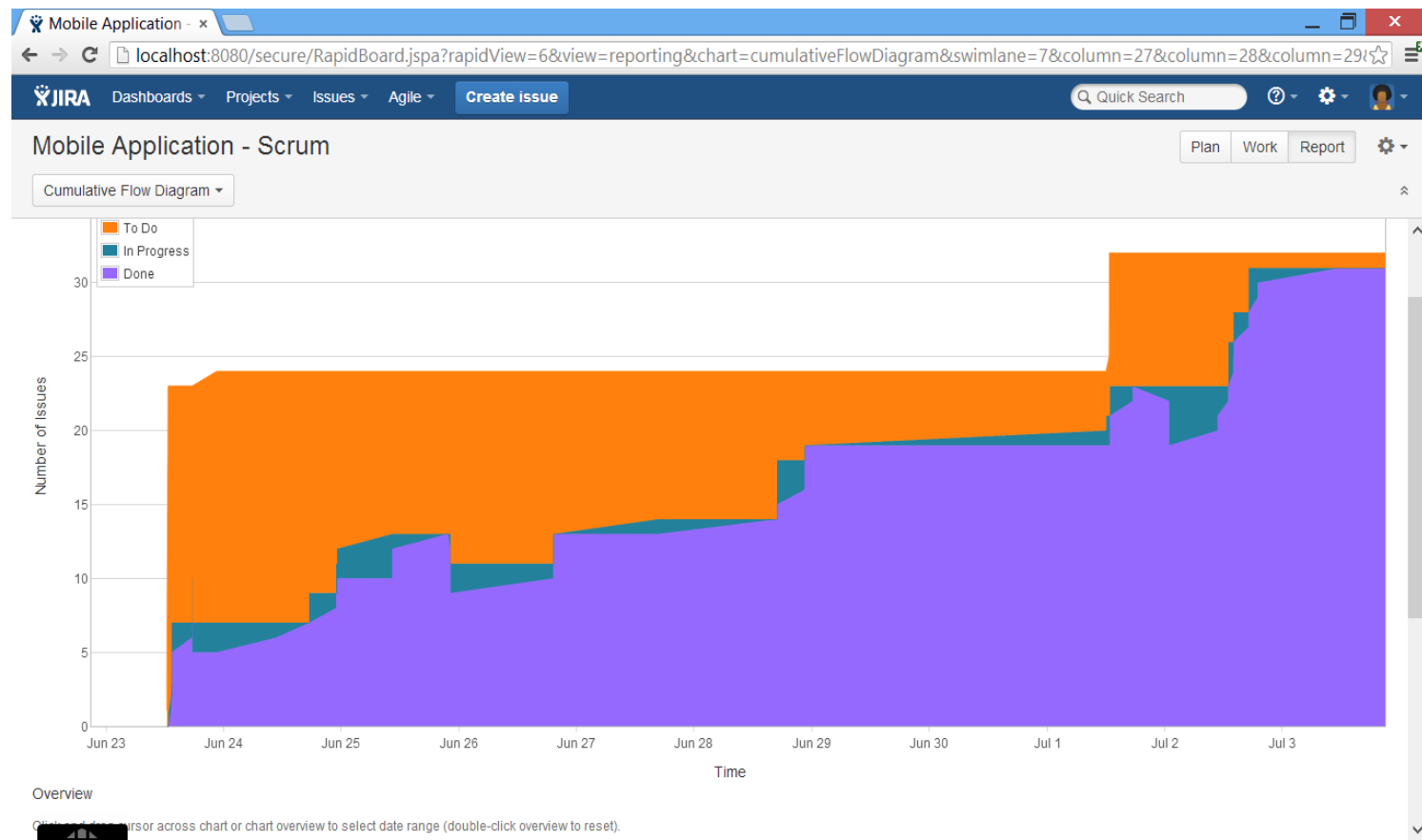
Details for SCRUM-45:

- Status: To Do
- Component/s: None
- Labels: None
- Affects Version/s: None
- Fix Version/s: None
- Epic: None
- Reporter: Iulia Gheorghe
- Assignee: Alex Stanciu
- Created: 01/Jul/13 12:43 PM

# REZULTATE EXPERIMENTALE

## – Agile/Scrum

### □ Diagrama stărilor



# REZULTATE EXPERIMENTALE

## – Agile/Scrum

### □ Diagrama valorii livrate



# REZULTATE EXPERIMENTALE

## – Agile/Scrum



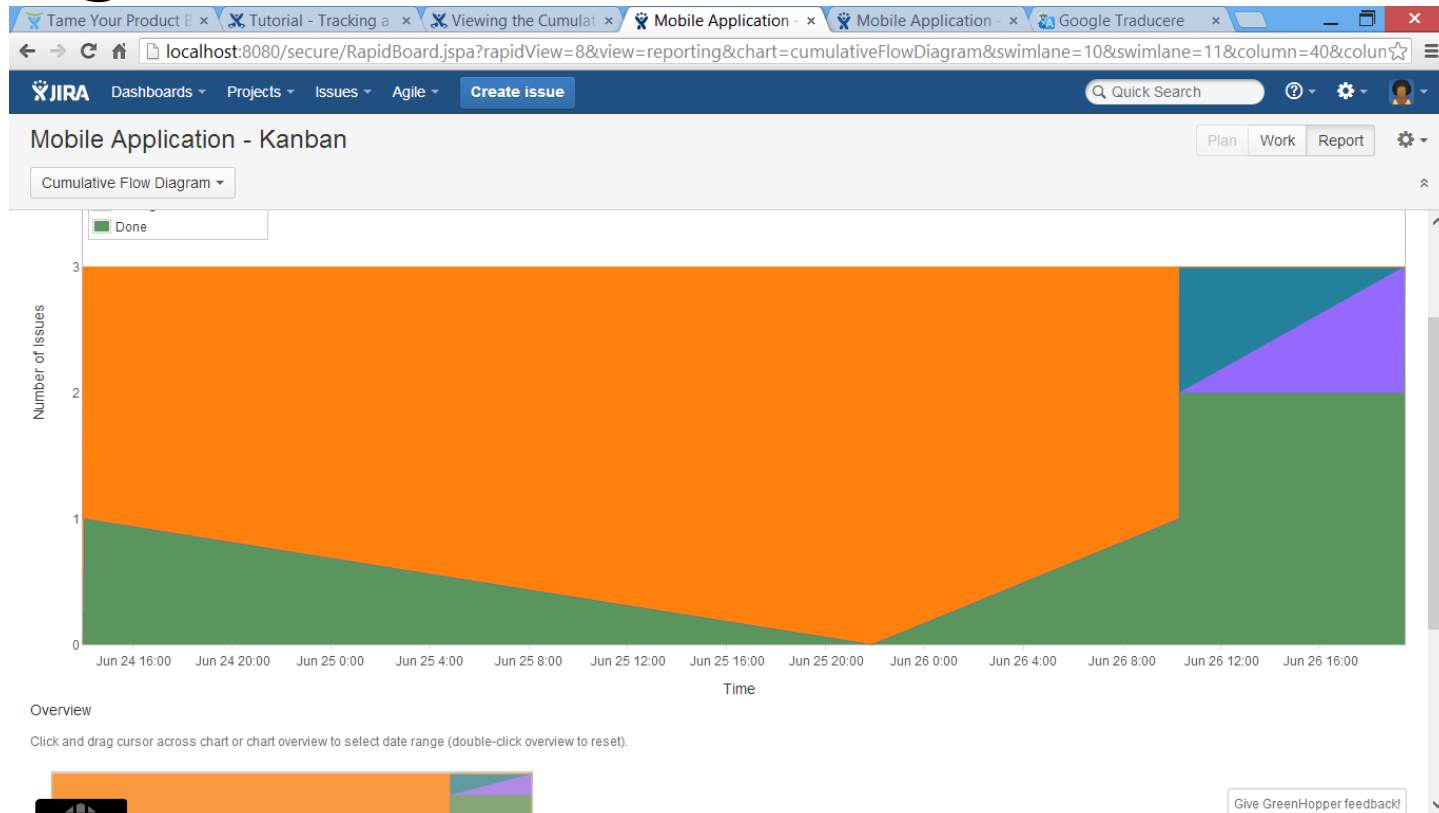
□ Story points pentru fiecare sprint

Sprint	Commitment	Completed
Sprint 1	0	0
Sprint 2	7	7
Sprint 3	3	3
Sprint 4	14	14
Sprint 1	17	17

# REZULTATE EXPERIMENTALE

## - Kanban

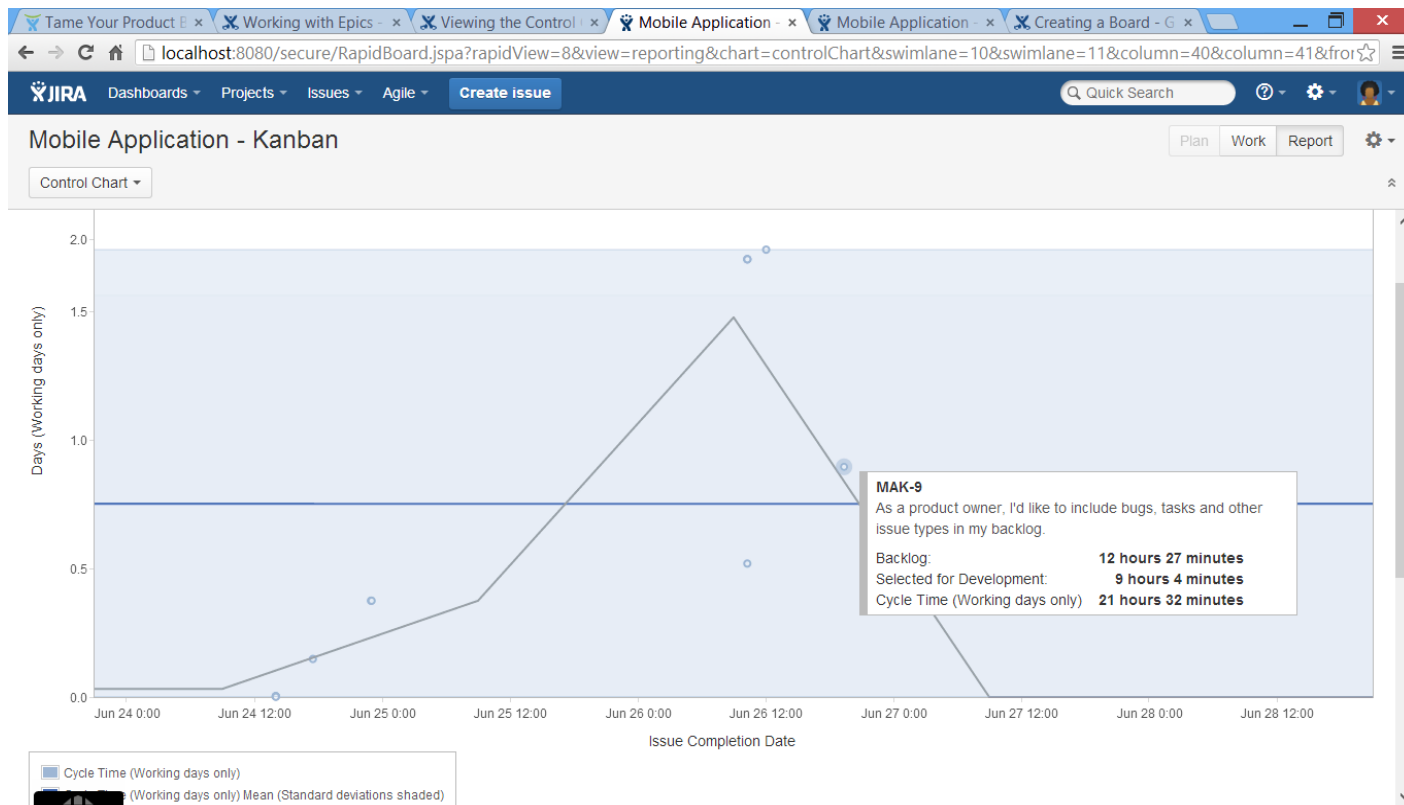
### ➤ Diagrama stărilor



# REZULTATE EXPERIMENTALE

## - Kanban

### □ Raportul de control



# CONCLUZII Scrum/Kanban aplicatie mobila



- În cazul abordării aplicației de gestiune mobile, este indicată folosirea metodei Kanban, deoarece tabla este mult simplificată comparativ cu Scrum Board, există puține task-uri de realizat, iar proiectul va fi finalizat într-un timp mult mai scurt și va avea costuri mai mici decât dacă am folosi Scrum Board. Dacă am implementa metoda Agile/Scrum, timpul alocat planificării ar fi mai mare și s-ar ‘pierde’ timp prețios și pentru ședințele specifice acestei metode.