Managementul Proiectelor

Curs 5

mihai.hulea@aut.utcluj.ro

Metodologii de Dezvoltare a Aplicatiilor Software

Introducere

- Ciclul de viata al unui produs software
- Ciclul de dezvoltare al unui produs software

Ciclul de devoltare al unui produs software

- Analiza cerintelor
- Realizarea specificatiilor
- Proiectare
- Realizare
- Testare
- Instalare
- Mentenanta

Metodologii de dezvoltare a produselor software

Cum efectuam activităţile indicate de ciclul de dezvoltare a programelor

Motodologii

- Complexe (formale si stricte);
- Simple (agile);

Exemple de modele de dezvoltare

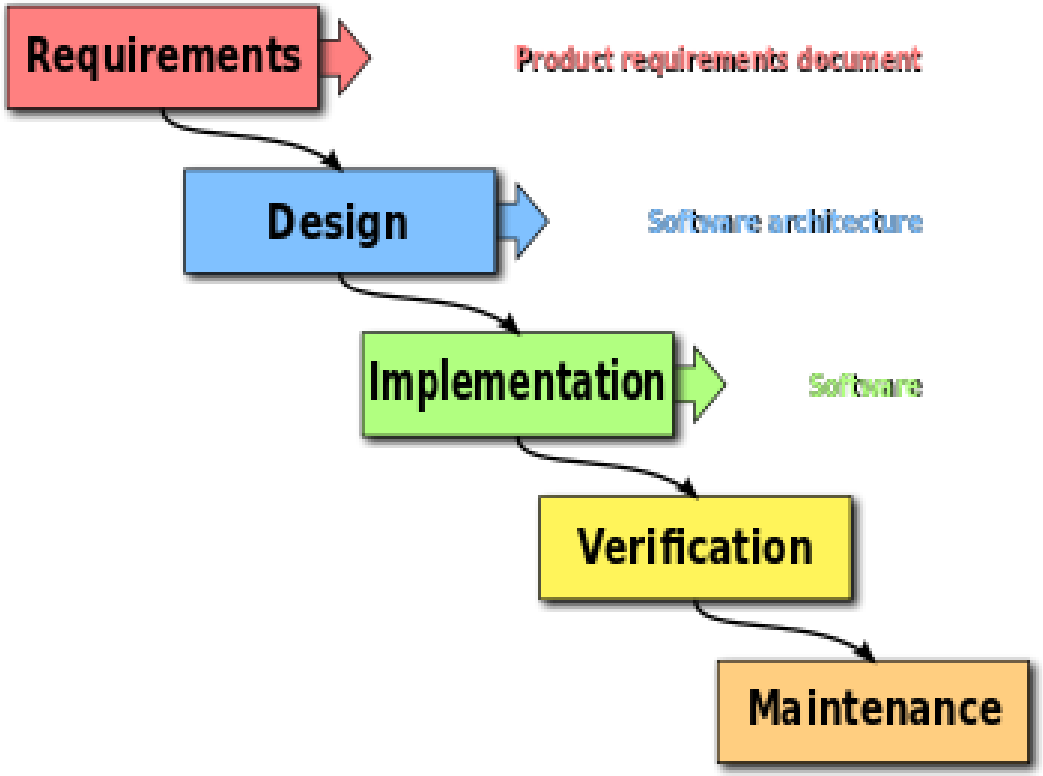
- Ad-hoc: descurca-te cum poţi
- Modelul în cascadă (cu feedback)
- Prototipizare
- Metode formale
- Modelul în spirală
- RUP (Rational Unied Process)
- XP (Extreme Programming)

Criterii pentru selectarea unei metodologii

- Buget
- Dimensiunea echipei
  - Metodologii agile = echipe mici
- Tehnologiile utilizate

- Unelte si tehnici
- Natura proiectului
- Procese existente in companie

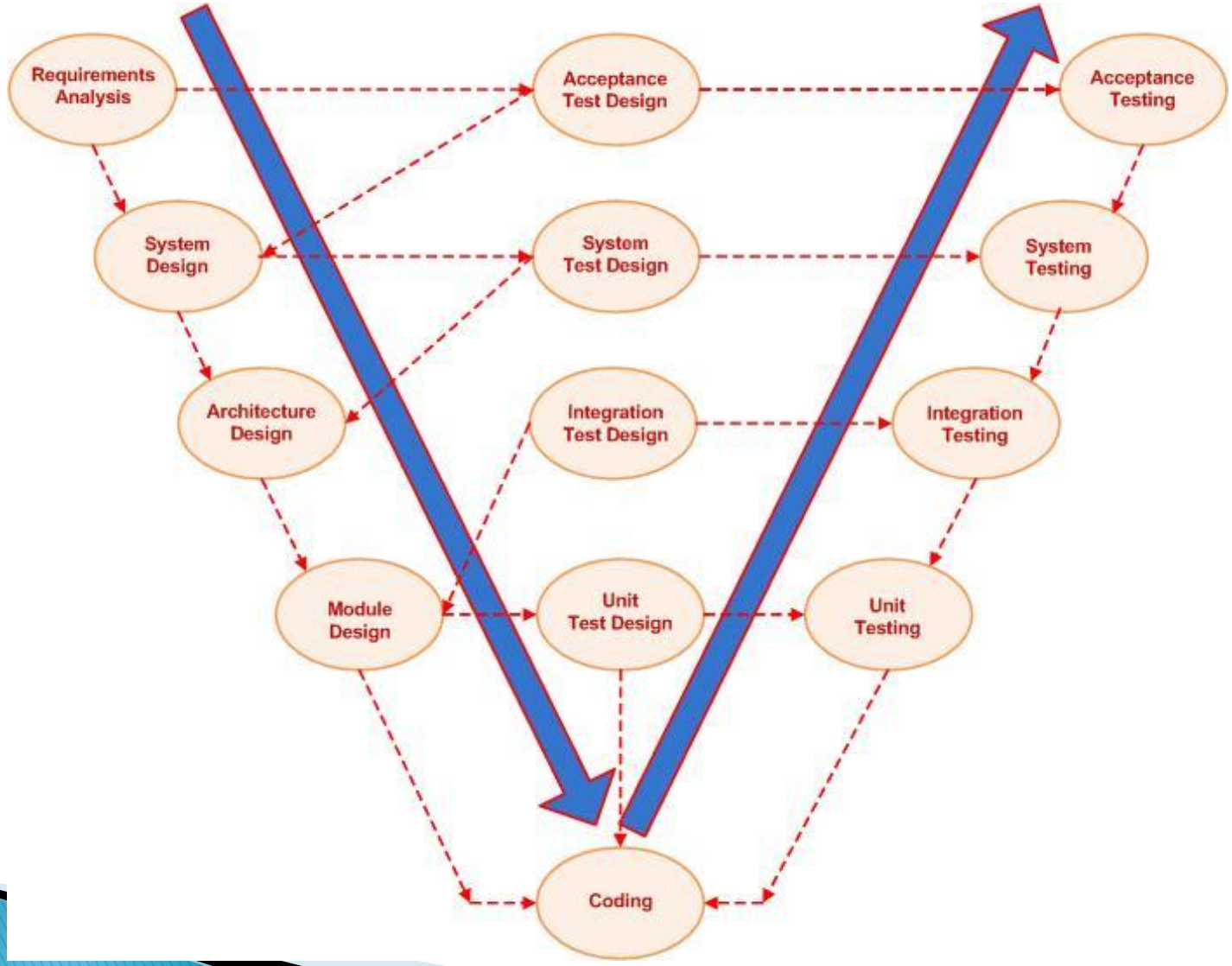
Metodologia Cascada



Metodologia Cascada

- Este o metodologie secventiala
- Utilizata de multe companiii mari
- Aplicabil pentru proiectele care sunt bine definiti de la inceput
- Iesirile unei faze reprezinta intrarile pentru faza urmatoare
- Avantaje
  - Este un model riguros
  - Pune accent pe documentare
  - Este simplu, usor de inteles si de urmarit
  - Usor de insusit de catre membrii echipei
- Dezavantaje
  - Nepractic in multe tipuri de proiecte
  - Ce se intampla daca clientul isi modifica cerintele pe parcursul proiectului?
  - Ce se intampla daca in faza de implementare se descompera ca o anumita componenta este difcil de implementat asa cum a fost proiectata initial ?
  - Pot aparea probleme noi pe parcursul dezvoltarii ce trebuies tratate

Metodologia V



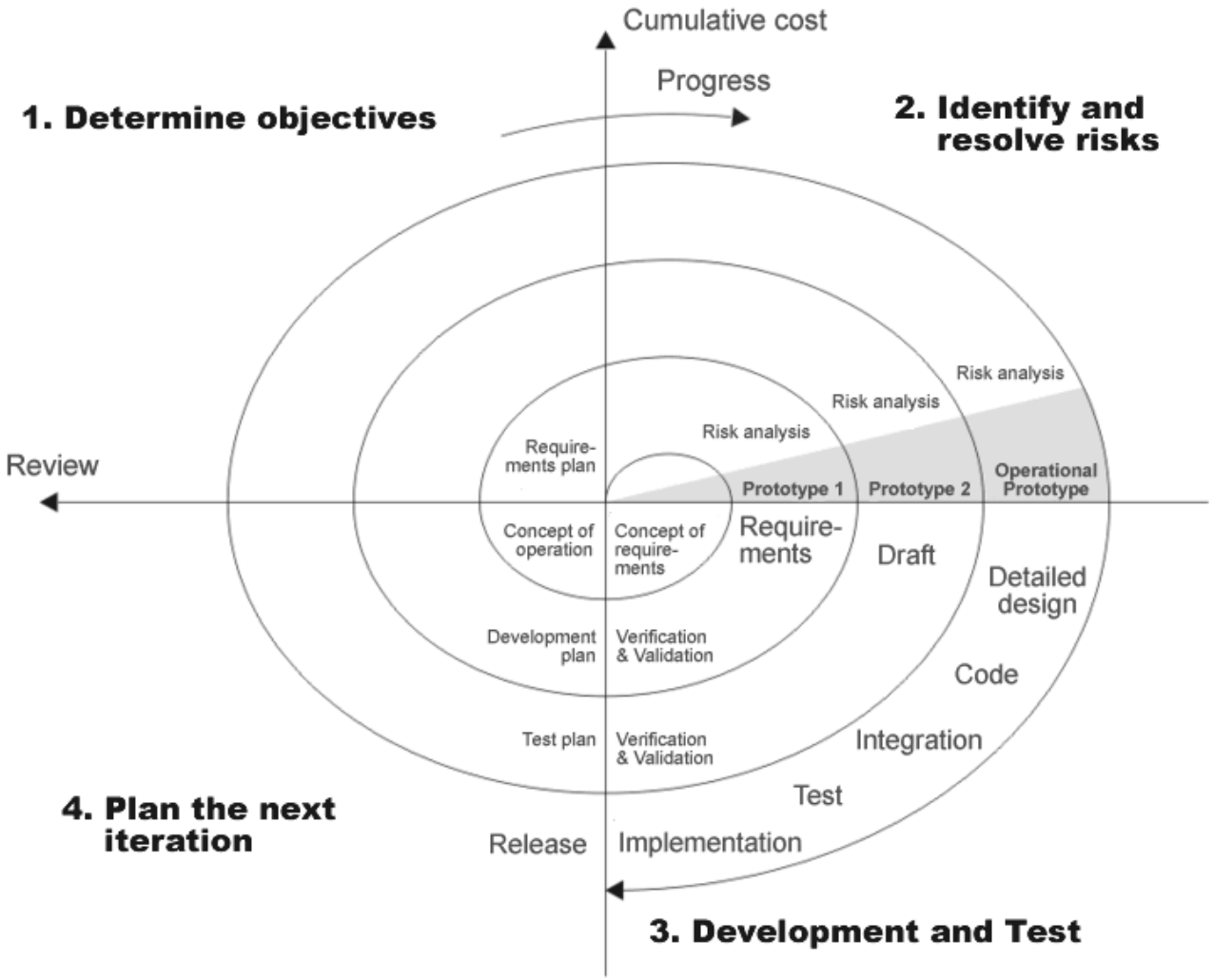
Metodologia V

- A fost dezvoltata in paralel in SUA si Germania
- Este un model rigid
- Testarea este activitate cheie in cadrul ciclului de dezvoltare
- Evita intoarcerile
- Durata de dezvoltare este lunga
- Produsul apare tarziu
- Partea superioara a V-ului implica utilizatorul final

Metodologia V

- Extensie a modelului cascada
- Parcurgerea fazelor nu este liniara
- Sunt definite 3 tipuri de faze
  - Fazele de verificare
  - Fazele de codare
  - Fazele de validare

Metodologia Spirala



Metodologia Spirala

- Cerintele sunt cunoscute
- Cerintele nu au necunoscute sau nu implica riscuri mari
- Cerintele nu se schimba pe parcursul dezvoltarii
- Solutiile tehnice si arhitectura adoptata este binecunoscuta de toti cei implicati
- Nu sunt constragneri de timp foarte stricte

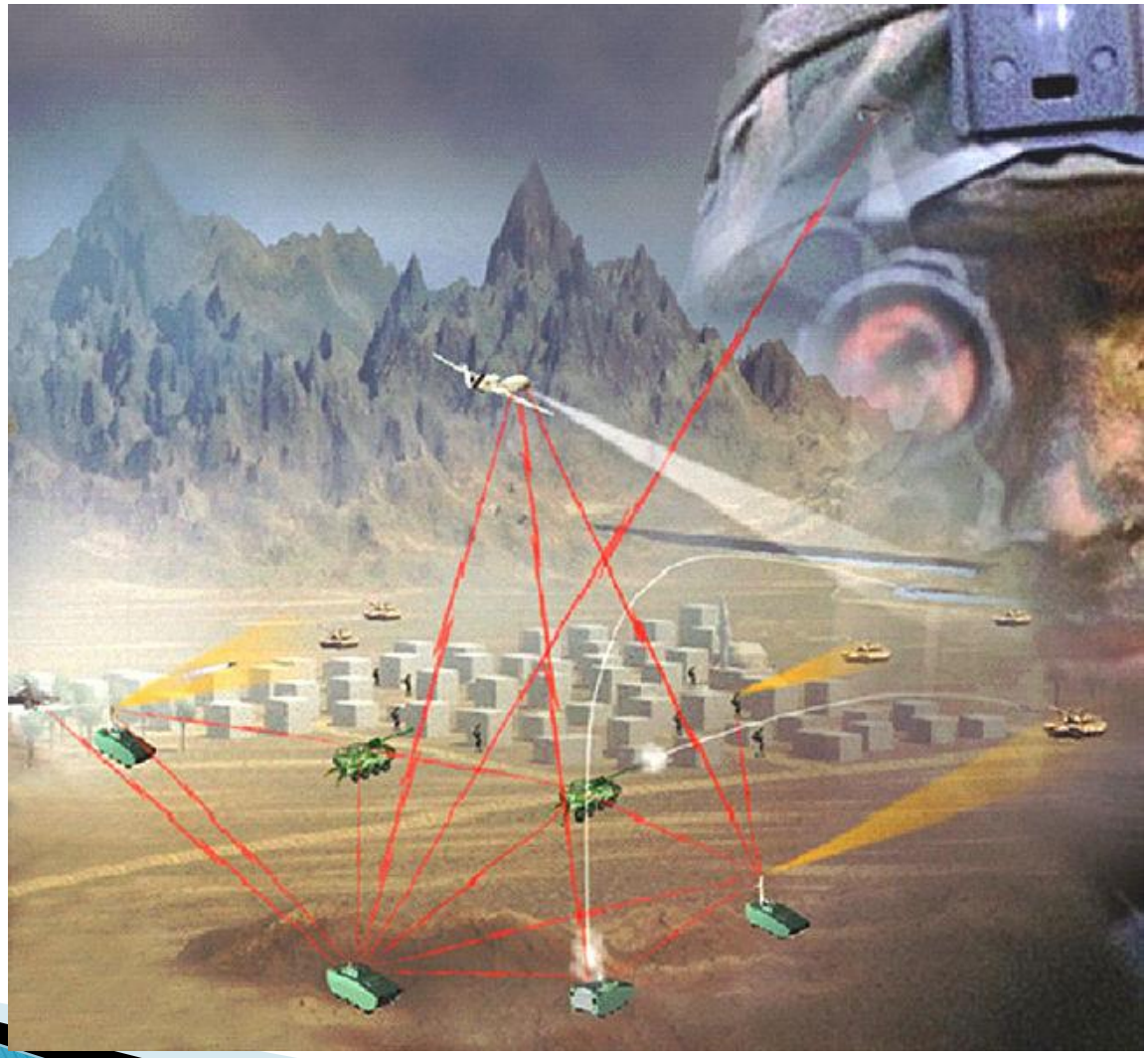
Metodologia Spirala

- Un model adoptat in multe proiecte IT
- Este un model iterativ
- Potrivit pentru proiecte mari, complexe
- Fiecare faza incepe cu analiza cerintelor si se incheie cu analiza de catre client a progreselor realizate in cadrul iteratiei
- Armata SUA a adoptat acest model pentru dezvoltarea sistemului Future Combat System (FCS)

Metodologia Spirala

- Cerintele pentru sistem sunt descrise in detaliu
- Este realizata o arhitectura preliminara a sistemului
- Este construit un prim prototip pe baza arhitecturii preliminare – in mod uzual un model la scara al produsului final
- Un nou prototip este realizat urmand procedurile:
  - Este evaluat primul prototip in termeni de calitati, defecte si riscuri
  - Sunt definite ceinrtele pentru noul prototip
  - Este planificat si construita arhitectura pentru noul prototip
  - Este construit si testat al doilea prototip

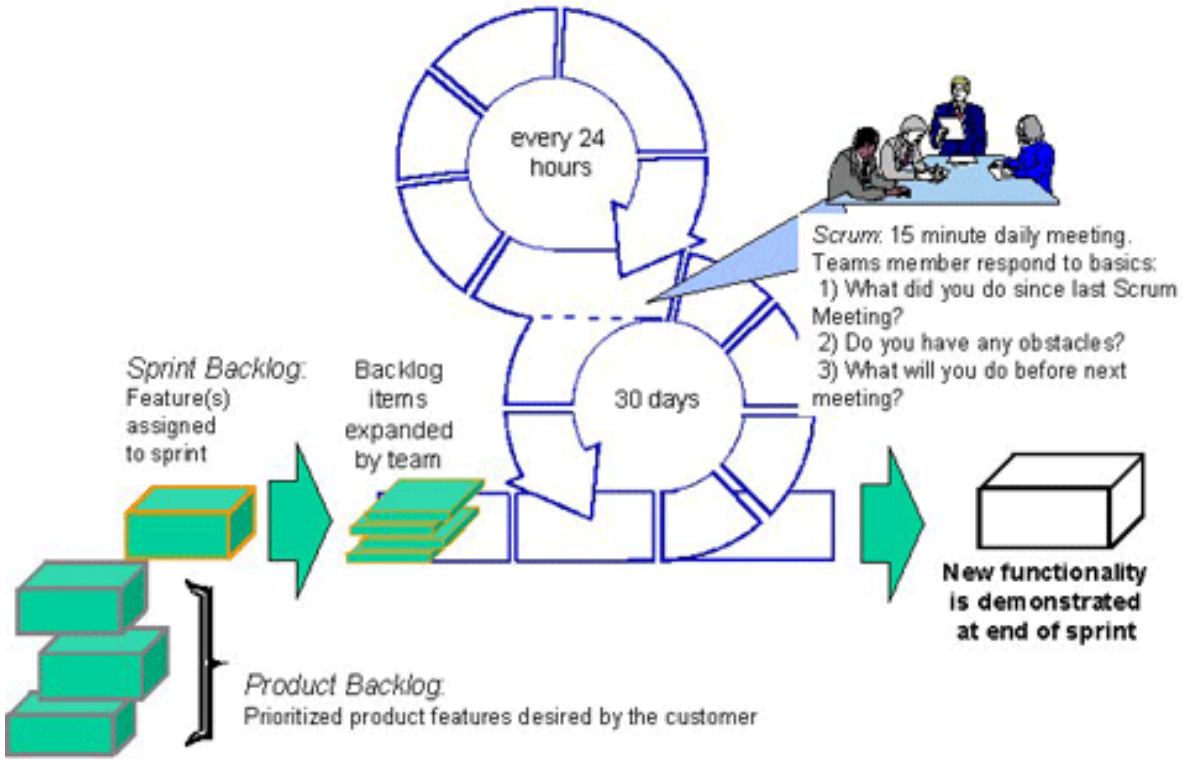
Metodologia Spirala



Metodologii agile

- Este un grup de metodologii de dezvoltare a aplicatiilor software
- Au la baza principiile iterative
- Minimizarea riscuri prin prin dezvoltarea in perioade scurte (iteratii)
- Durata unei iteratii de la 1 la 4 saptamani
- Fiecare iteratie cuprinde toate sarcinile
- Fiecare iteratie poate fi privita ca un miniproiect
- Rezultatul fiecarei iteratii este o noua versiune a produsului
- Se pune accent pe comunicarea in timp real
- Colaborare intre programatori si client
- In general sunt echipe mici de dezvoltare
- In raport cu alte metode produc foarte putina documentatie scrisa (nediscipinate)
- Metode agile: XP, Scrum

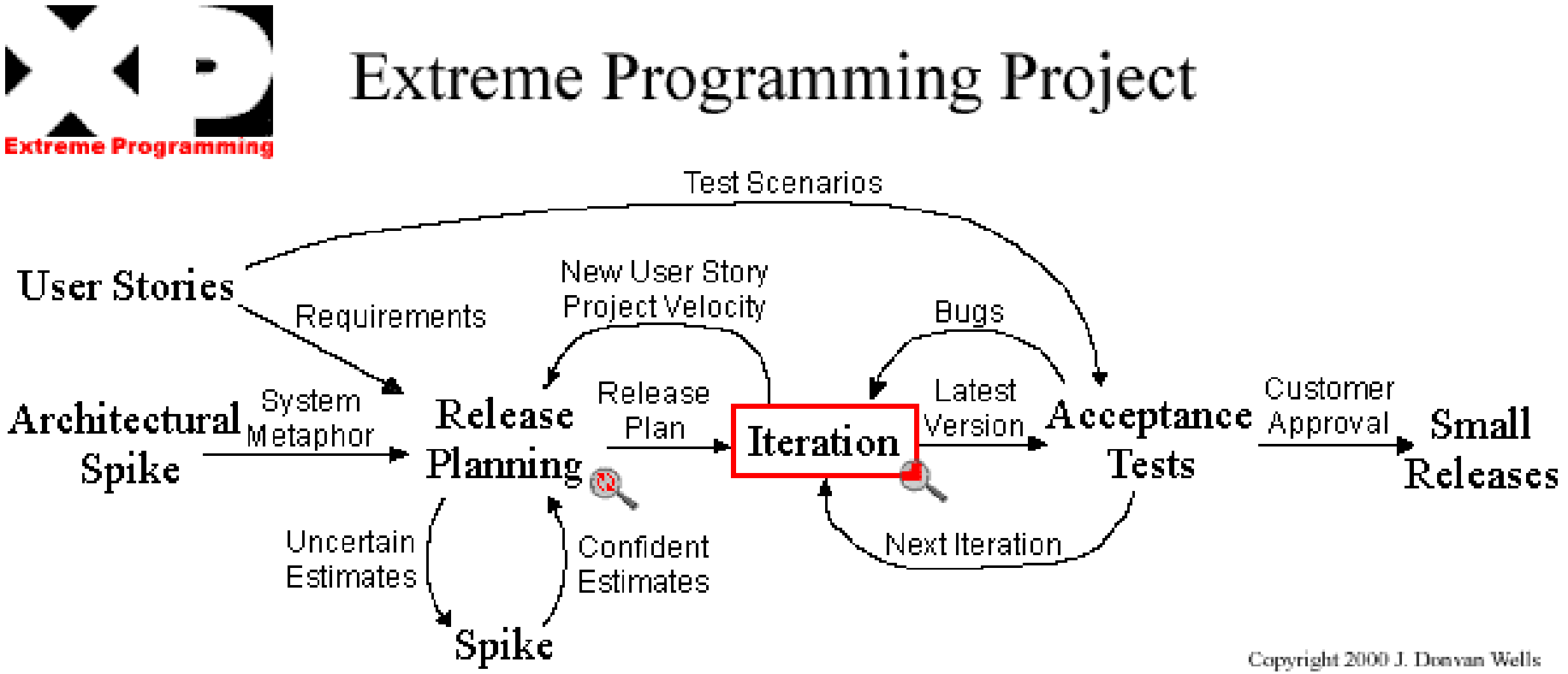
Metodologia Scrum



Metodologia Scrum

- Proiectul este divizat in pachet mici asignate membrilor echipei
- Echipele de proiect sunt in general mici
- Partile implicate sunt divizate in
  - Partea ineteresata de rezultat
  - Partea care dezvolta
- Scrum Master este liderul partii care dezvolta
- Sprint
  - Un sprint este o etapa sau un ciclu in dezvoltarea produsului finalizata printr-o versiune
  - Sprintul dureaza 1 – 3 saptamani
- Etapele sunt:
  - Faza de inititializare in care sunt evaluate rezultate sprintului anteriori si se planifica sprintul ce urmeaza
  - Faza de productie in care se implementeaza si testeaza
  - Faza de inchidere in care rezultatele sunt prezentate clientului
- Un proiect este realizat in mai multe sprinturi de preferat de durate egale
- Pot fi setate borne la fiecare n sprinturi pentru evaluare
- Finalizarea ultimului sprin este echivalent cu finalizarea produslui si livrarea lui
- Au loc sedinte zilnice conduse de Scrum Master
  - Ce ai lucrat ieri?
  - Ce probleme ai intampinat?
  - Ce vei lucra azi?

Metodologia XP



Metodologia XP

- Extreme Programming(XP)
- XP descrie 4 principale activitai
  - Codare – principala activitate
  - Testare – orice modul implementat trebuie testa
  - Ascultare(comunicare) – programatorul trebuie sa comunice cu clientul pentru a intelege necesitatile acestuia
  - Design – realizarea unei arhitecturi corecte a sistemului va eficientiza sistemul si va reduce dependentele care nu sunt necesare intre diferitele module ale sistemului
- Se preteaza pentru proiecte cu cerinte dinamice sau cele care nu sunt bine definite de la inceput
- Trebuie sa existe un parteneriat intre client si programatori
- Nu genereaza foarte multa documentatie

Metodologia XP

- Echipa de dezvoltare nu are o structură ierarhica. Fiecare contribuie la proiect folosind maximul din cunoştinţele sale.
- Scrierea de cod este activitatea cea mai importantă.
- Proiectul este în mintea tuturor programatorilor din echipa, nu în documentaţii, modele sau rapoarte.
- La orice moment, un reprezentant al clientului este disponibil pentru clarificarea cerinţelor.
- Codul se scrie cât mai simplu.
- Se scrie cod de test întâi.
- Daca apare necesitatea rescrierii sau aruncării de cod, aceasta se face fără milă.
- Modificările aduse codului sunt integrate continuu (de câteva ori pe zi).
- Se programează în echipă (programare în perechi). Echipele se schimbă la sfârşitul unei iteraţii (1-2 săptămâni).
- Se lucrează 40 de ore pe săptămână, fără ore suplimentare.

Metodologia RUP

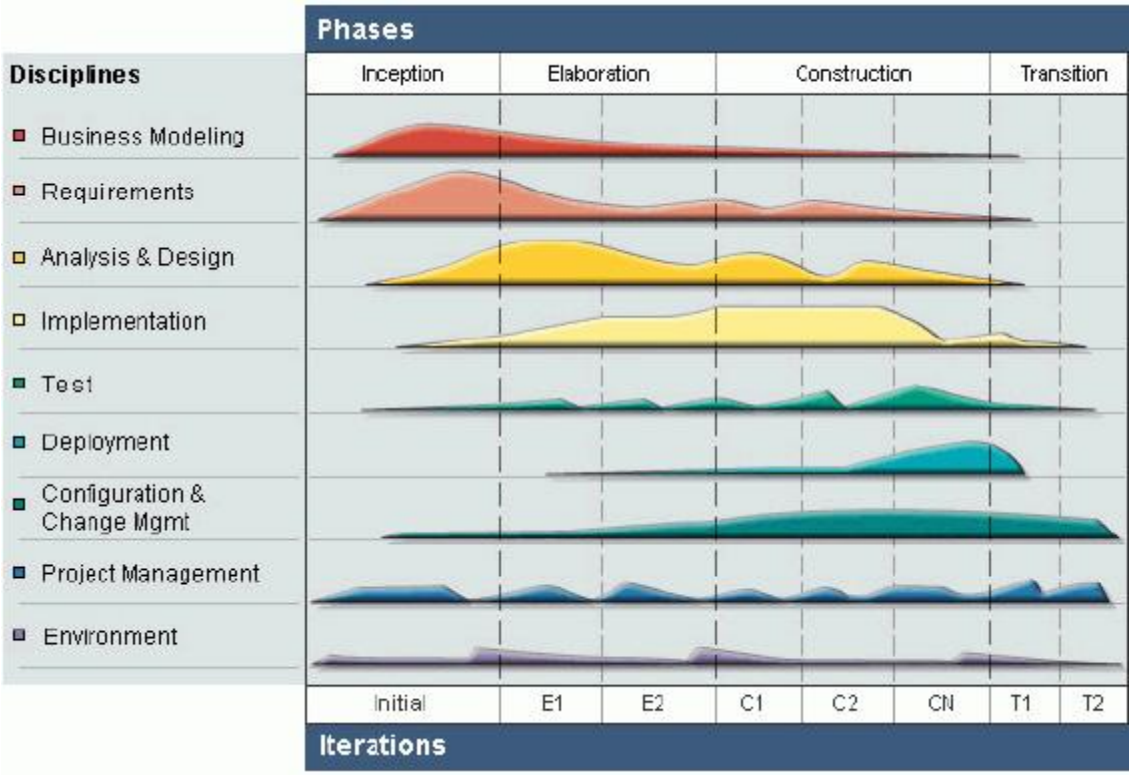
- **Rational Unified Process**
- Metoda iterativa dezvoltata de catre Rational Software
- Este o colectie de procese ce descriu ciclul de dezvoltare al produselor software
- Pune acent pe descriere proceselor sub forma unor diagrame – in contrast cu celelate modele
- Utilizeaza UML ca limbaj de descriere
- IBM ofera unelte pentru automatizarea proceselor descrise de RUP

Metodologia RUP

- RUP defineste 6 principii de urmat
  - Dezvolta software-ul in mod iterative
  - Gestioneaza cerintele
  - Utilizeaza arhitecturi bazate pe component
  - Modeleaza in mod visual aplicatiile
  - Verificia calitatea software-ului
  - Gestioneaza schimbarile in software

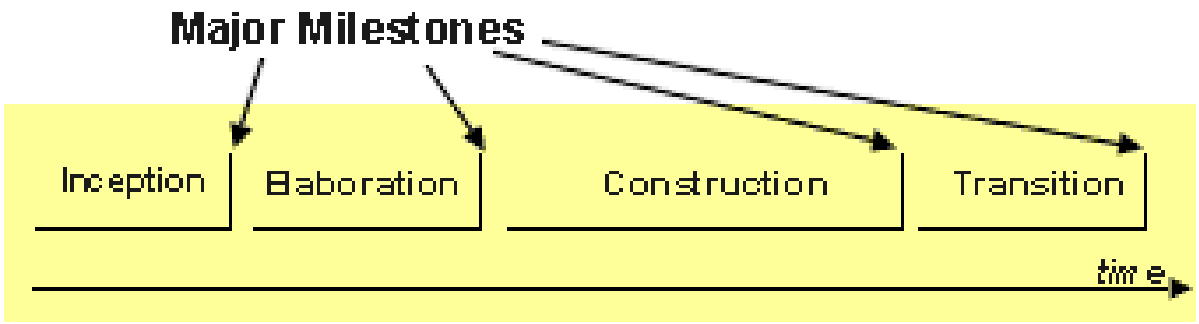
Metodologia RUP

- Procesele definite de RUP
  - Dimensiune timp, dimensiune statica



Metodologia RUP (cont)

- Ciclul de dezvoltare al aplicatiei este descompus in cicluri, fiecare ciclu este compusa din faze
- Fazele sunt:
  - Initializare
  - Elaborare
  - Constructie
  - Tranzitie



RUP – Initializarea

- Un document de viziune
- Un model Use-case initial (10-20% complet)
- Un plan de proiect cuprinzand fazele si iteratiile
- O analiza a riscurilor
- Un model de bussines daca e necesar
- Unul sau mai multe prototipuri

RUP – Elaborare

- Cea mai critica faza a proiectului
- Cele mai importante componente ale sistemului sunt dezvoltate acum
- Un model use-case (80% complet)
- Capturarea cerintelor suplimentare
- Arhitectura generala a sistemului
- Un prototip executabil
- Modelul de bussinis si riscurile revizuite
- Planul de realizare al proiectului
- Un manual preliminar de utilizare (optional)

RUP – Constructie

- Restul componentelor nerealizate in faza anterioara constuite si integrate in produs
- Toate componentele sunt testate
- Rezultatul trebuie sa fie un produs pregatit pentru a fi livrat utilizatorului
- Este realizat manualul de utilizare

RUP – Tranzitie

- Produsul este instalat\livrat utiliatorului
- Utilizatorul testeaza produsul
- Defectele sunt raportate si fixate
- Aceasta faza poate fi foarte simpla sau foarte complexa depinzand de tipul produsului (ex. Aplicatie desktop, aplicatie pe mai multe niveluri)

RUP – Iteratii

- Fiecare faza din RUP poate fi descompusa in iteratii
- Avanatajele metodelor iterative
  - Riscurile sun gestionate mai bine
  - Schimbarile sunt mai usor de controlat
  - Calitate mai buna

RUP – Unelte

- **Rational Requisite®Pro**--Keeps the entire development team updated and on track throughout the application development process by making requirements easy to write, communicate and change.
- **Rational ClearQuest**™--A Windows and Web-based change-request management product that enables project teams to track and manage all change activities that occur throughout the development lifecycle.
- **Rational Rose 98**--The world's leading visual modeling tool for business process modeling, requirements analysis, and component architecture design.
- **Rational SoDA**--Automates the production of documentation for the entire software development process, dramatically reducing documentation time and costs.
- **Rational Purify**®--A run-time error checking tool for application and component software developers programming in C/C++; helps detect memory errors.
- **Rational Visual Quantify**™--An advanced performance profiling tool for application and component software developers programming in C++, Visual Basic, and Java; helps eliminate performance bottlenecks.
- **Rational Visual PureCoverage**™ --Automatically pinpoints areas of code not exercised in testing so developers can thoroughly, efficiently and effectively test their applications.
- **Rational TeamTest**--Creates, maintains and executes automated functional tests, allowing you to thoroughly test your code and determine if your software meets requirements and performs as expected.
- **Rational PerformanceStudio**™--An easy-to-use, accurate and scalable tool that measures and predicts the performance of client/server and Web systems.
- **Rational ClearCase**®--Market-leading software configuration management tool, giving project managers the power to track the evolution of every software development project.

Referinte

- https://en.wikipedia.org/wiki/Waterfall_model
- https://en.wikipedia.org/wiki/Spiral_model

Sfarsit