Fundamente de Inginerie Software

Cap. 2

Ciclul de Viață a Unui Sistem Software. Modele Specifice Fazelor Ciclului de Viată.

Conf.Dr.Ing. Dan Pescaru

Textbooks: Maciaszek "Practical Software Engineering", 2005, Cap. 1

Sursă: http://www.comp.mq.edu.au/books/pse/

2009

Fazele ciclului de viaţă

- *Fazele ciclului de viață a unui* ***sistem software (Maciaszek'05)****:*
  - Analiza cerințelor
  - Proiectarea sistemului
  - Implementarea
  - Integrarea şi instalarea la beneficiar
  - Operarea şi întreținerea

Analiza cerinţelor

- Determinarea cerințelor – una din provocările cele mai dificile
- Specificarea cerințelor – Unified Modeling Language (UML)
- Computer Assisted Software Engineering (CASE)
- Documentația de specificare a cerințelor:
  - Serviciile oferite de sistem (ce trebuie să facă sistemul)
  - Constrângerile sistemului
- Asigurarea calității software-ului
  - Parcurgeri şi inspecții

Proiectarea Sistemului (1)

- Proiectarea sistemului constă în (Sommerville'04):
  - O descriere a structurii sistemului de implementat
  - Datele care sunt prelucrate în sistem
  - Interfețele între componentele sistemului
  - Algoritmii utilizați (doar în anumite situații)
- În practică distincția între analiză şi proiectare nu este foarte clară
  - Modelele de viață sunt iterative şi incrementale
  - Acelaşi limbaj de modelare (UML) este utilizat şi la analiză

Proiectarea Sistemului (2)

- Proiectarea detaliată (adaugă detalii modelului rezultat din analiza cerințelor)
- Proiectarea arhitecturală
- Gestiunea relaționării parților aflate în diverse stadii de dezvoltare – cerințe, proiect sau segmente de cod – (Traceability Management)

Implementarea

- Implementarea este în mare parte programare, dar
  - Proiectul este sub-specificat (în zona algoritmilor)
  - Extra-proiectare înainte de codificare
- Un programator este un “inginer de componente”
- Programarea este o “inginerie în circuit”
- Integrated Development Environments (IDEs)
  - Generare de cod (forward engineered) din model (proiect) +reverse
  - Testare şi depanare
- Revizuirea codului (prin treceri şi inspecții)
- Testare bazată pe execuție (observarea comportamentului)
  - Testarea conforma cu specificațiile (black-box testing)
  - Testarea conforma cu codul (white-box testing) – se urmăresc căi de execuție

Integrarea şi Instalarea

- In pasul de integrare se asamblează aplicația din setul de componente implementate şi testate în prealabil
  - Dificil de distins față de:
    - Implementare (integrare continuă la “agile development”)
    - Testare (“integration testing”)
  - Condusă de proiectarea arhitecturală a sistemului
- Instalarea reprezintă înmânarea sistemului funcțional beneficiarilor pentru utilizarea în producție
  - Softul este instalat în diverse versiuni
  - Fiecare versiune este precedată de testarea de sistem (dezvoltator – alpha-testing) şi testarea de acceptare (beneficiar – beta-testing)
  - Training pentru beneficiar
  - Documentație de utilizare

Testarea

- **Stub** – o piesă de cod care simulează comportamentul unei componente neimplementate încă
  - Se utilizează în testarea “top-down”
  - Problemă: dependențele circulare. **Big-bang testing** nu este o soluție
- **Driver** – o piesă de cod care conduce integrarea a.î. versiunea increment (build) poate primi datele şi contextul care ar fi furnizate de componentele neimplementate încă
  - Se utilizează în testarea “ bottom-up”
- **Teste de suport (test harness)** – teste care utilizează “stubs” şi “drivers” (utile doar în timpul integrării)

Operarea şi Întreţinerea

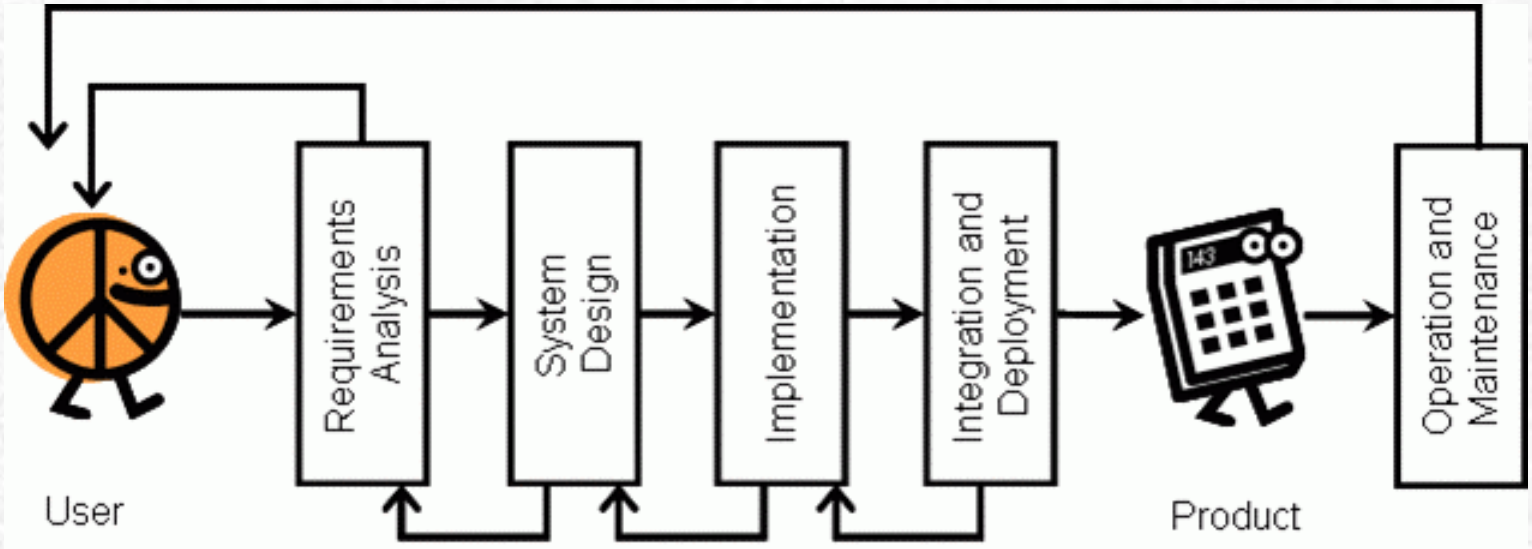
- **Operarea** semnifică acea fază a ciclului de viață în care un produs software este utilizat în munca de zi-cu-zi înlocuind sistemul precedent
- Startul **Operării** coincide cu începerea procesului de **Mentenanță**
  - Corectivă (“de casă”)
  - Adaptivă
  - Perfectivă
- Sisteme “moştenite” (**Legacy Systems**)

Variaţiile ciclului de viaţă

- Ciclul de viață poate diferi funcție de:
  - Experiența, abilitățile şi cunoştințele membrilor echipei de dezvoltare
  - Gradul de cunoaştere şi experiența în afacerea vizată de sistem
  - Tipul domeniului aplicației
  - Schimbările din mediul afacerii
  - Schimbările din interiorul afacerii
  - Dimensiunea proiectului

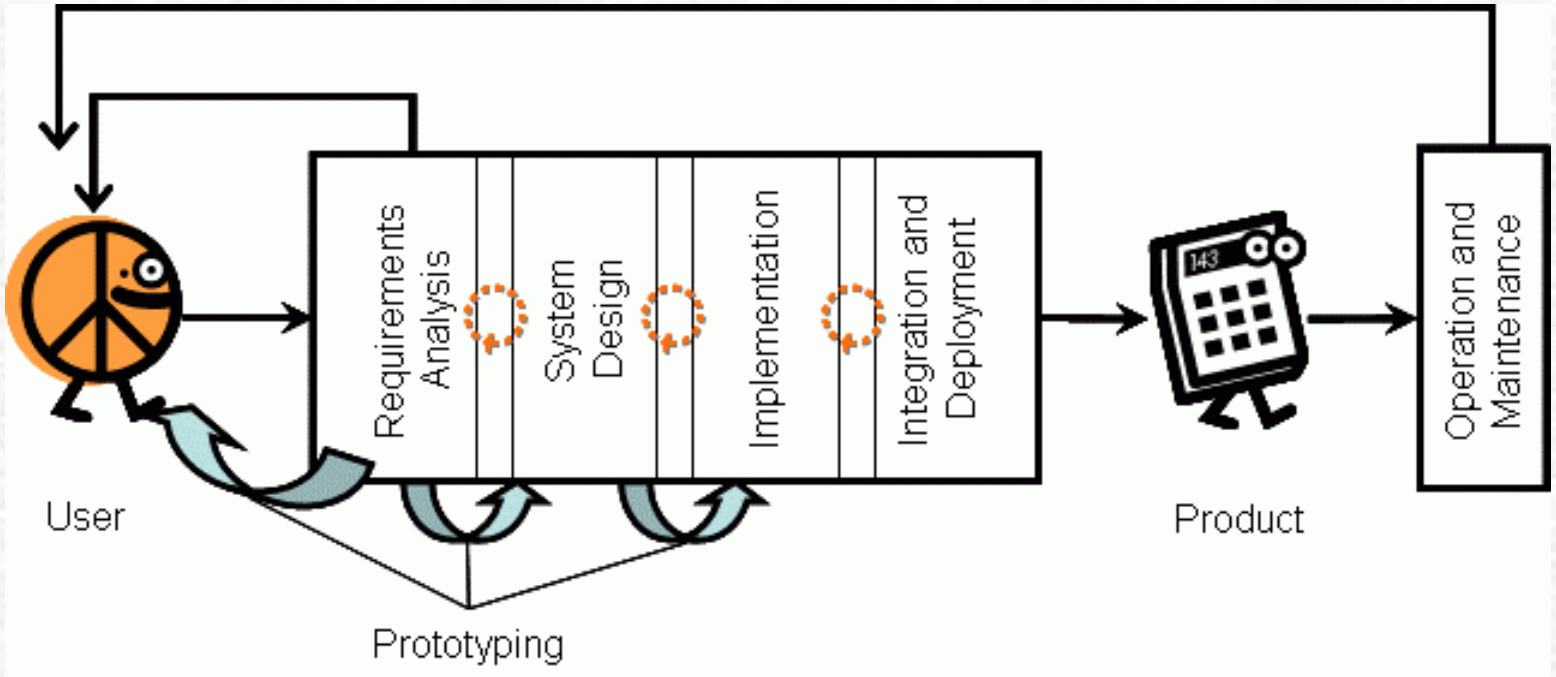
Modelul în cascadă cu reacţie

- Modelul în cascadă cu reacție (Waterfall lifecycle with feedback) (Maciaszek'05)



Modelul în cascadă cu suprapuneri

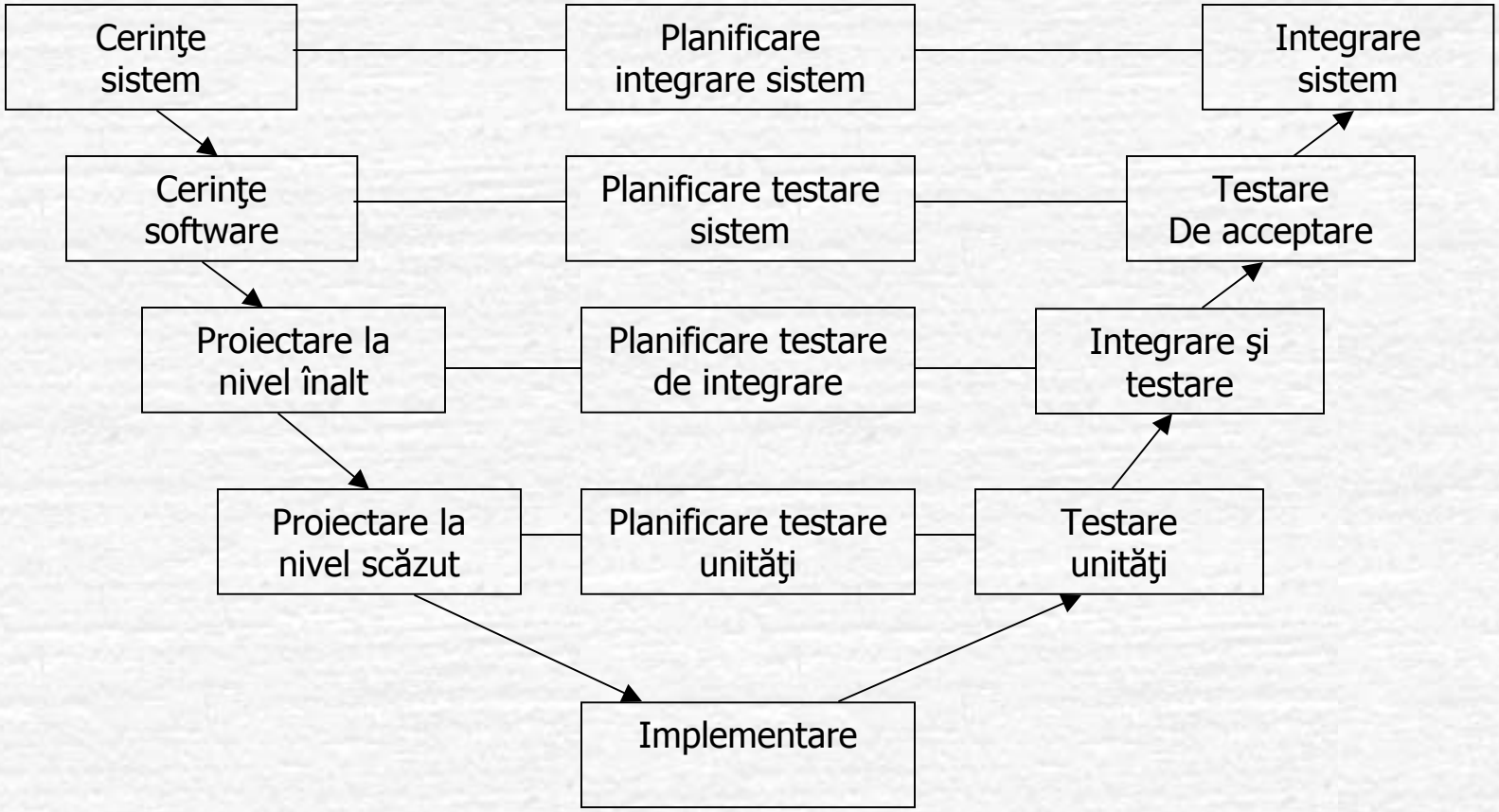
- Modelul în cascadă cu reacție, suprapuneri şi prototipuri (Waterfall with feedback, overlaps, and prototypes)



Modelele în cascadă

- Ambele modele au aproximativ aceleaşi caracteristici
- Avantaje:
  - Simplu şi uşor de utilizat
  - Uşor de gestionat datorită rigidității
  - Fazele şi procesele sunt terminate pe rând (uşor de urmărit)
  - Bun pentru proiectele mici unde cerințele sunt bine înțelese încă de la început
- Dezavantaje
  - Modificarea cerințelor este foarte greu de gestionat
  - Nu sprijină dezvoltarea orientată pe obiecte
  - Nu se produce prototipuri executabile decât foarte târziu

Modelul în "V"


Cerinţe sistem
Planificare integrare sistem
Integrare sistem
Cerinţe software
Planificare testare sistem
Testare De acceptare
Proiectare la nivel înalt
Planificare testare de integrare
Integrare şi testare
Proiectare la nivel scăzut
Planificare testare unităţi
Testare unităţi
Implementare

Modelul în “V”

- Modelul în “V” presupune un ciclul de viață secvențial
- Planificările se fac o dată cu parcurgerea primei ramuri
- Avantaje:
  - Simplu şi uşor de utilizat, mai ales pentru proiecte mici
  - Fiecare fază are lucruri specifice, controlabile, de livrat
  - Mia bun decât modelul în cascadă deoarece planul de testele este făcut încă de la început
- Dezavantaje
  - Rigid, greu de introdus modificări dacă apar
  - Nu produce prototipuri timpurii

Ciclul de viaţă iterativ

- **Iterația** în inginerie sofware este o repetiție a unui proces cu scopul de adăuga funcționalități unui produs software
- Ciclul de viață bazat pe iterații presupune creşteri succesive (**increments)** – versiuni îmbunătățite sau extinse ale produsului la sfârşitul fiecărei iterații
- Ciclul de viață iterativ presupune versiuni succesive îmbunătățite (**builds)** – sub formă de cod executabil livrabil la încheierea fiecărei iterații
- Dezavantaj: faze rigide şi care nu se suprapun

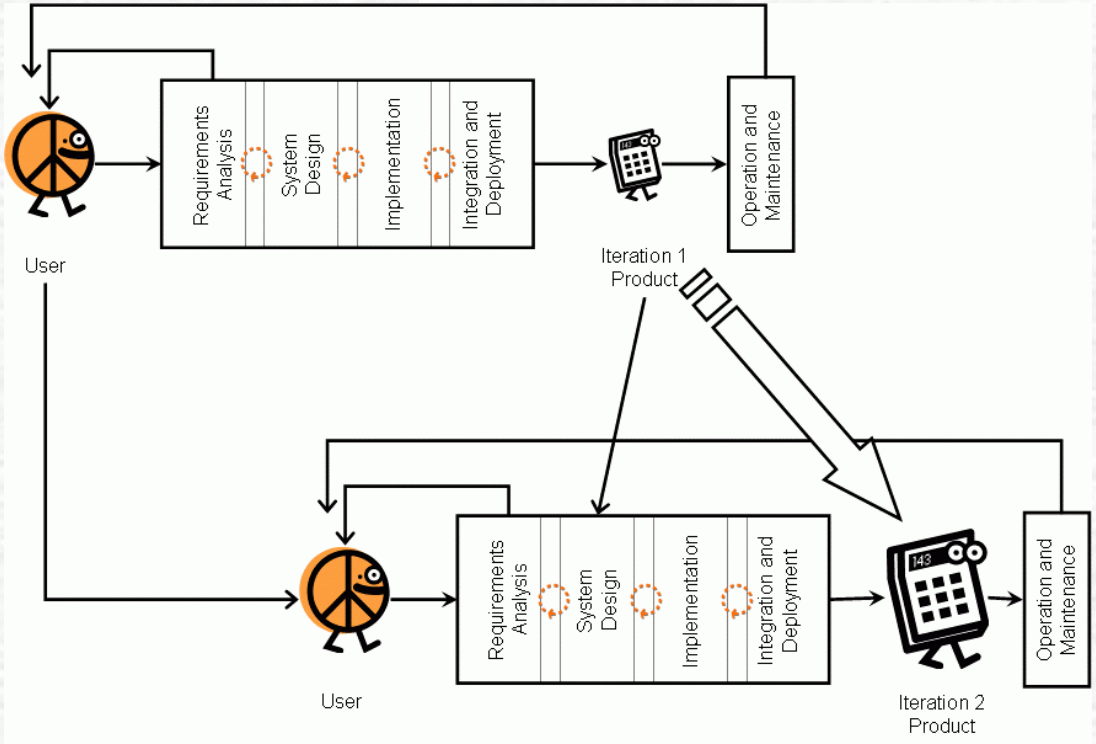
Ciclul de viaţă iterativ. Modele

- Ciclul de viață iterativ presupune **iterații scurte** între variante succesive (zile sau săptămâni, nu luni)

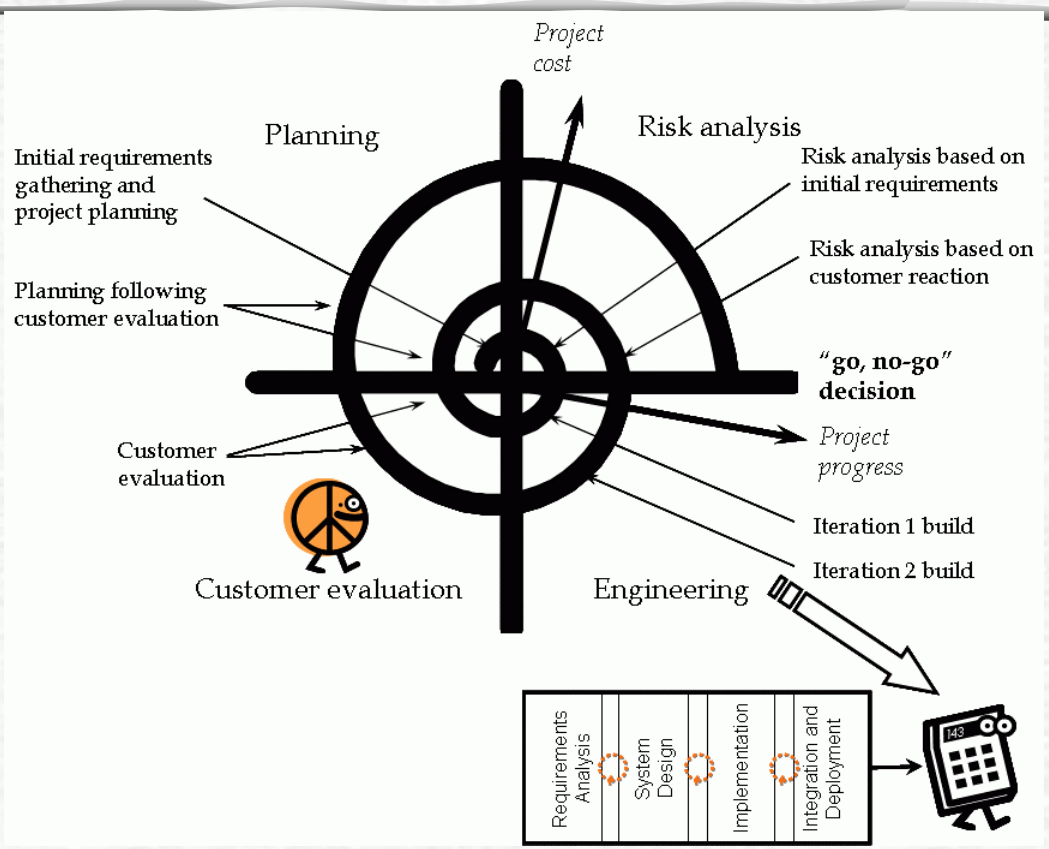
- Modele:
  - Spirală (Boehm, 1988)
  - IBM Rational Unified Process (RUP) (IBM, 2003)
  - Model Driven Architecture (MDA) (OMG, 2003)
  - Agile lifecycle with short cycles (Agile Alliance, 2001)

Ciclul de viaţă iterativ. Schemă

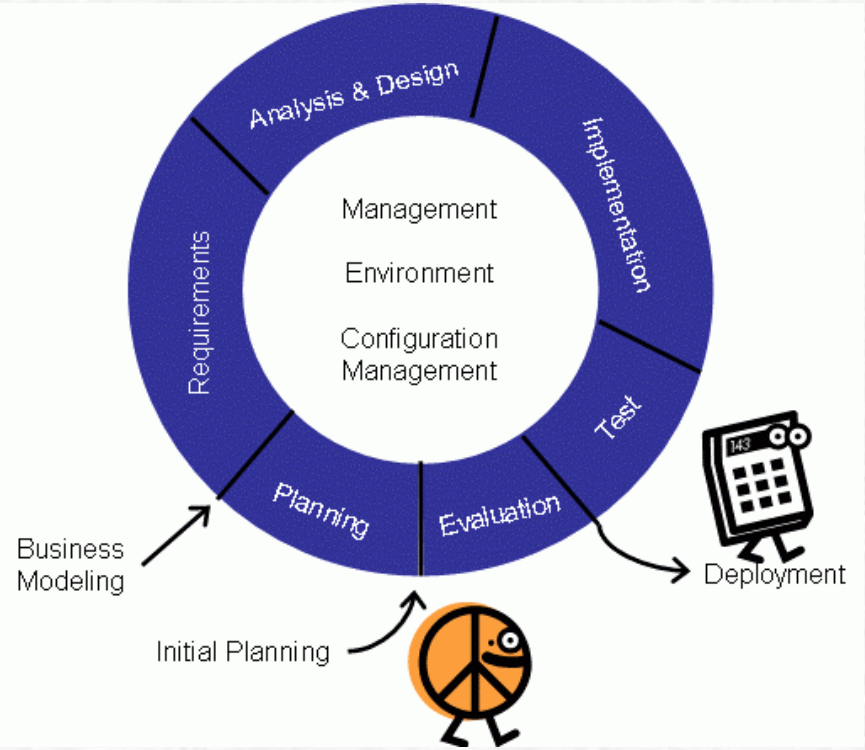
- Ciclul de viață iterativ cu variante succesive



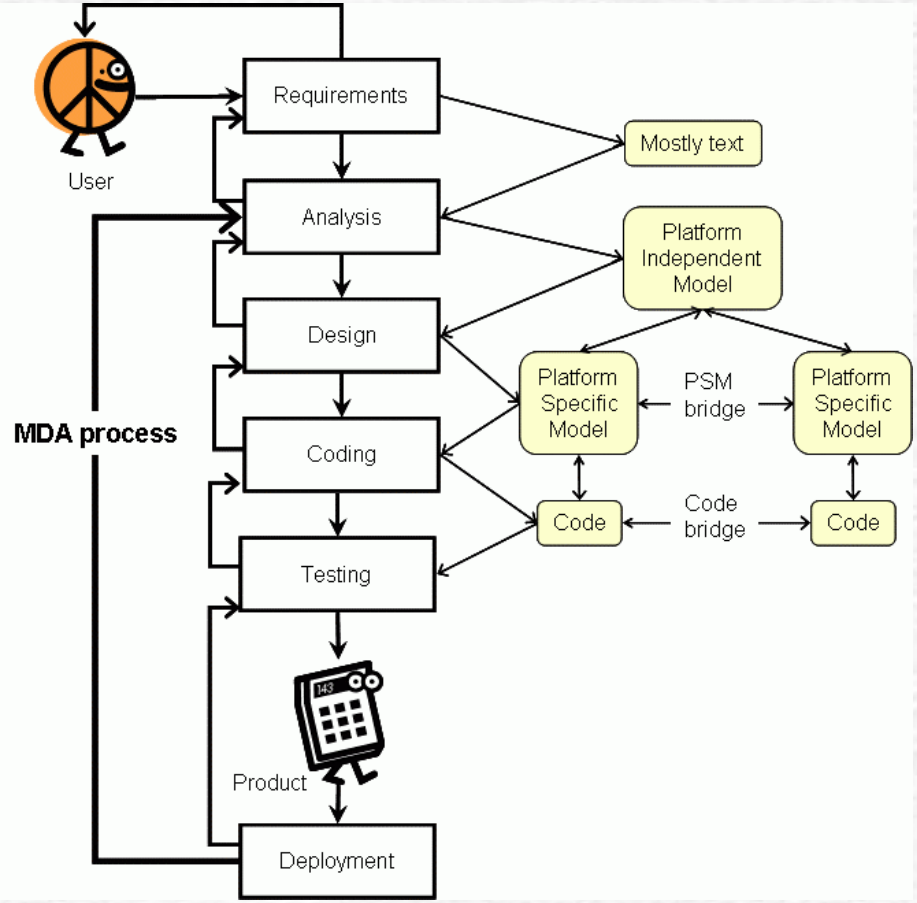
Modelul Spirală



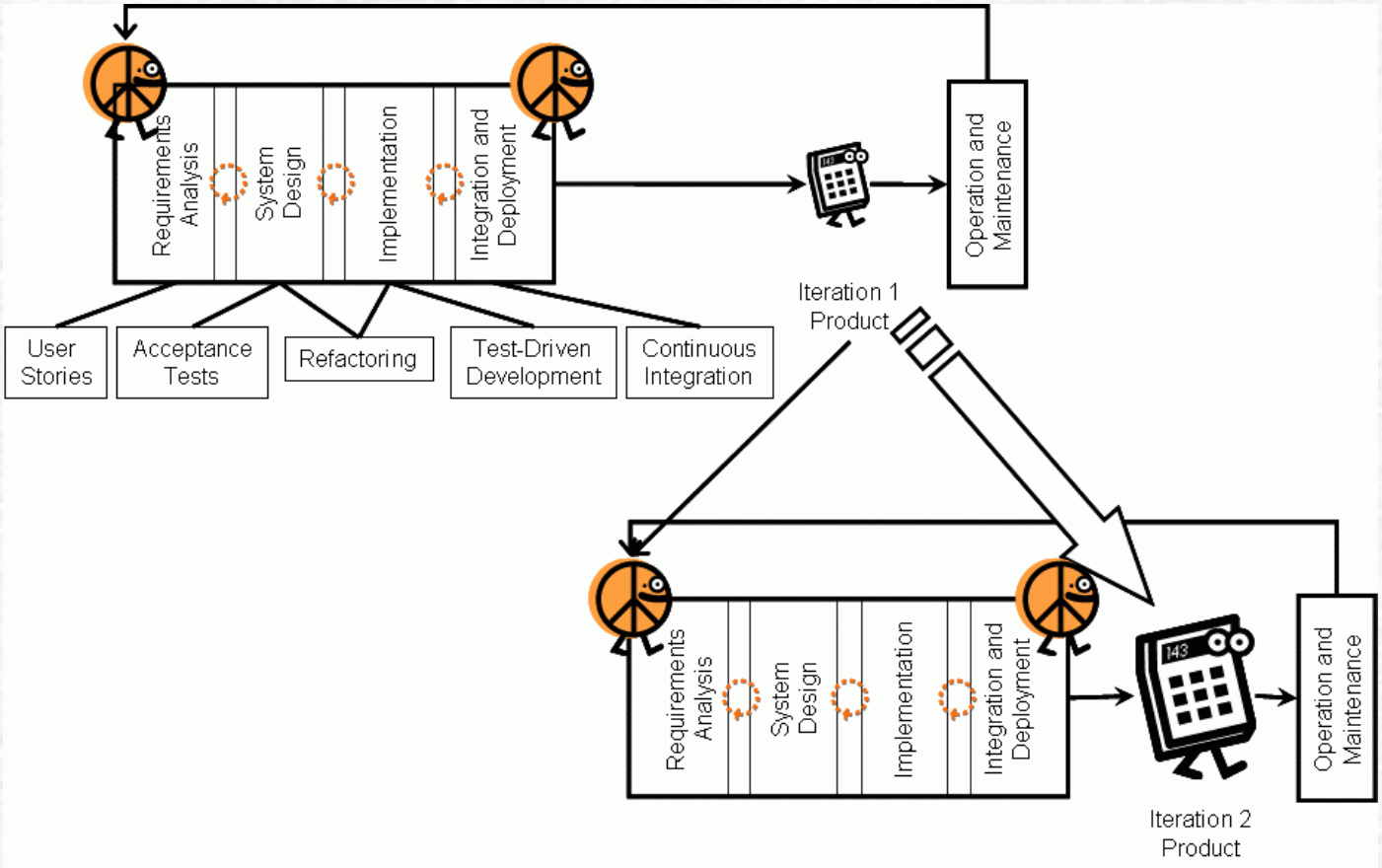
Modelul RUP


Analysis & Design
Implementation
Requirements
Management
Environment
Configuration Management
Test
Planning
Evaluation
Business Modeling
Deployment
Initial Planning

Modelul MDA


User
Requirements
Analysis
Design
Coding
Testing
Deployment
MDA process
Product
Mostly text
Platform Independent Model
Platform Specific Model
PSM bridge
Platform Specific Model
Code
Code bridge
Code

Modelul Agile



Practici Extreme Programming (XP)

- *Reacție fină*
  - *Programare în perechi*
  - *Planificarea "jocului"*
  - *Dezvoltare condusă de teste*
  - *Echipă completă*
- *Proces Continu*
  - *Integrare continuă*
  - *Refactorizare (îmbunătățirea proiectului)*
  - *Variante în paşi mici*
- *Cadru adecvat pentru programator*
  - *Atmosferă relaxată*
- *Practici comune echipei*
  - *Codare standard*
  - *Drepturi comune asupra codului*
  - *Proiectare simplă*
  - *Denumiri explicite*

*XP Develoment. Courtesy of wikipedia

Concluzii

- Stadiile procesului de dezvoltare a software-ului sunt cunoscute ca faze ale ciclului de viață software
- Fazele ciclului de viață sunt: analiza cerințelor, proiectarea sistemului, implementarea, integrarea şi instalarea, operarea şi mentenanța
- Modelele ciclului de viață pot fi împărțite în:
  - Modele cascadă cu reacție
  - Modele iterative cu variante succesive
- Modelele în cascadă nu sunt adecvate proceselor moderne de dezvoltare
- Cele mai reprezentative patru modele iterative sunt: Spirală, Rational Unified Process (RUP), Model Driven Architecture (MDA) şi Agile.