Dezvoltarea agila - 1

Prof. univ. dr. ing. Florica Moldoveanu

Ingineria programelor

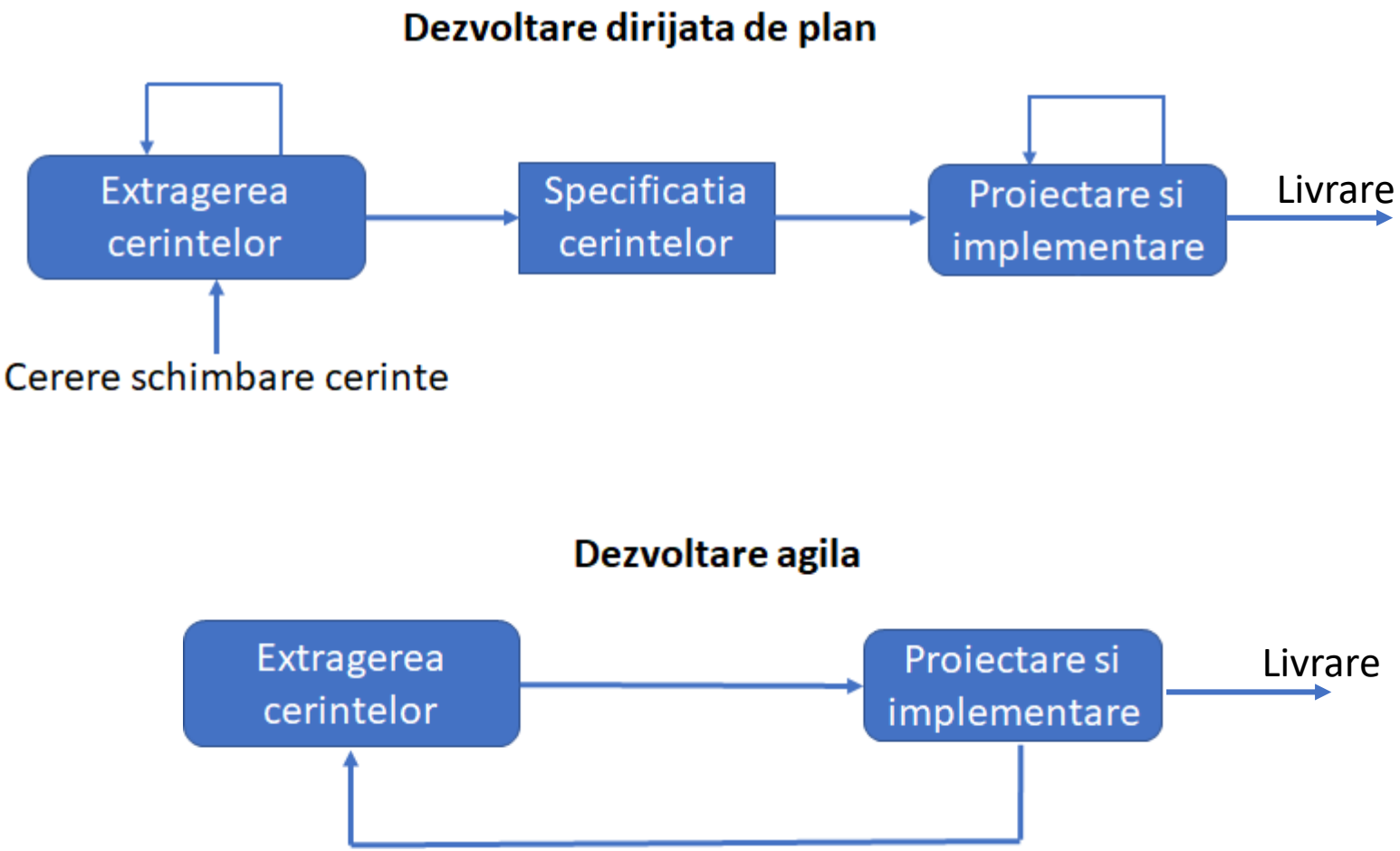
UPB, Automatică şi Calculatoare

2018-2019

Dezvoltarea agila (1)

- Dezvoltarea si livrarea rapida → aspecte importante ale proceselor software in prezent
  - ✓ Cerintele software se schimba rapid si este dificil de produs un set de cerinte stabile
  - ✓ Produsul software trebuie sa evolueze rapid pentru a reflecta schimbarile in cerinte
- **Dezvoltarea dirijata de plan** este esentiala pentru anumite tipuri de sisteme dar nu satisface cerintele unei dezvoltari rapide, adaptate la schimbarea frecventa a cerintelor
- Metodele agile au aparut la finalul anilor 1990, din necesitatea de a reduce drastic timpul de livrare pentru produsele software.
- **Dezvoltare agila**:
  - ✓ Specificarea, proiectarea si implementarea sunt intercalate
  - ✓ Sistemul este dezvoltat printr-o serie de incremente cu partile interesate implicate in specificarea si evaluarea versiunilor
  - ✓ Livrari frecvente
  - ✓ Utilizare intensiva de instrumente de dezvoltare (ex. Instrumente de testare automata)
  - ✓ Documentatie minimala – focalizare pe cod functional

Dezvoltarea agila (2)



Dezvoltarea agila (3)

❖ **Dezvoltarea dirijata de plan**

- Dezvoltare in etape separate, cu rezultatul fiecarei etape planificat in avans
- Nu neaparat dezvoltarea cascada – dezvoltarea iterativa dirijata de plan este posibila (ex. RUP)
- Iteratiile sunt planificate la inceputul procesului de dezvoltare

❖ **Dezvoltarea agila**

- Specificarea, proiectarea, implementarea si testarea sunt intrepatrunse
- Rezultatul procesului de dezvoltare este decis printr-un proces de negociere in timpul procesului de dezvoltare.

Metodologii de dezvoltare agile

PRINCIPII GENERALE

Implicarea clientului	**Clientul trebuie sa fie puternic implicat in procesul de dezvoltare. Rolul lui este de a furniza si prioritiza noi cerinte si de a evalua iteratiile dezvoltarii.**
Livrarea incrementala	**Produsul software este dezvoltat in incremente, clientul specificand cerintele care trebuie sa fie implementate in fiecare increment.**
Oameni, nu procese	**Abilitatile echipei de dezvoltare trebuie sa fie recunoscute si exploatate. Membrii echipei trebuie lasati sa-si dezvolte propriul stil de lucru, fara sa li se impuna procese prescriptive.**
Inglobeaza schimbarea	**Se asteapta schimbarea cerintelor, de aceea sistemul trebuie proiectat pentru a fi usor adaptat la schimbari.**
Mentine simplitatea	**Focalizare pe simplitate atat in produsul software dezvoltat cat si in procesul de dezvoltare. Elimina complexitatea oricand este posibil.**

Agile Manifesto – cele 4 valori fundamentale

- Metodologiile agile au aparut din necesitatea de a elimina din dezvoltarea de software activitatile care conduc la intarzierea livrarii de software operational.
- Termenul “dezvoltare agila” - derivat din **Agile Manifesto**, publicat in 2001 de un grup din care au facut parte creatorii unor metodologii existente la acea vreme: Scrum, Extreme Programming (XP), Dynamic Systems Development Method (DSDM) si Crystal.
- Agile Manifesto a stabilit un set de valori și principii generale comune metodologiilor agile existente la momentul respectiv. Sunt definite **4 valori fundamentale** care permit echipelor de dezvoltare software sa fie performante:
  - **Indivizi si interactiuni** in loc de procese si instrumente
  - **Software functional** in loc de documentatie cuprinzatoare
  - **Colaborarea cu clientul** in loc de negociere contractuala
  - **Raspunsul la schimbari** in loc de urmarirea unui plan

Agile Manifesto – cele 12 principii (1)

1. Prioritatea maxima o are satisfacerea clientului prin furnizarea rapida si continua de software functional.
2. Schimbarile in cerinte sunt binevenite, chiar si tarziu in dezvoltare, in avantajul clientului.
3. Livrarea frecventa de software functional, la cateva saptamani sau luni, preferabil la intervale mai scurte.
4. Dezvoltatorii si expertii din domeniul aplicatiei trebuie sa lucreze impreuna zilnic pe toata durata proiectului.
5. Construieste proiecte în jurul unor persoane motivate. Dă-le mediul și sprijinul de care au nevoie, și ai încredere în ei pentru a obtine rezultatul.

   Echipele agile sunt alcatuite din oameni care doresc sa lucreze impreuna colaborativ si sa invete unul de la altul. Oamenii sunt principalul factor de succes in dezvoltarea de software.

6. Cea mai eficace metoda de transmitere a informatiilor este comunicarea directa in echipa.
7. Software-ul functional este principala masura a progresului.

Agile Manifesto – cele 12 principii (2)

8. Procesele agile promoveaza dezvoltarea durabila. Sponsorii, dezvoltatorii si utilizatorii trebuie sa fie capabili sa mentina un ritm constant pe timp nelimitat.

   Pentru a obtine rezultate de calitate, oamenii nu trebuie fortati sa munceasca 12 ore/zi in permanenta.

9. Atenția continua la excelenta tehnica si o buna proiectare maresc agilitatea.

   Practici: refactoring - mentinerea unui cod de calitate si test-driven development - software-ul este functional in permanenta. Se folosesc ghiduri de codare si uneori ghiduri de modelare.

10. Simplitatea – arta de a maximiza cantitatea de munca nefacuta – este esentiala.

    Dezvoltatorii agili se concentreaza pe activitatile cu valoare mare, care maximizeaza ROI in IT, automatizand sau renuntand la munca nefolositoare.

11. Cele mai bune arhitecturi, cerinte si proiectari rezulta din echipe care se auto-organizeaza.

    Agile Model Driven Development (AMDD) și Test Driven Design (TDD) sunt abordările principale din cadrul comunității agile, pentru producerea de arhitecturi eficiente, cerințe, și modele de calitate.

12. La intervale regulate echipa analizeaza cum poate sa devina mai eficienta apoi isi ajusteaza corespunzator comportamentul.

Metode agile

❖ **Cand sunt aplicate metodele agile in dezvoltarea de software?**

- Atunci cand se doreste dezvoltarea rapida a unui produs software, fara clarificarea tuturor cerintelor de la inceput, acestea urmand sa fie extrase si implementate pe baza feedback-ului utilizatorilor la versiunile livrate pe parcurs.
- Atunci cand se doreste dezvoltarea personalizata a produsului software, cu angajarea clara a clientului de a deveni implicat in procesul de dezvoltare si exista putine reguli si reglementari externe care sa afecteze produsul software.

Care metodologie este mai potrivita pentru un proiect, dezvoltarea liniara dirijata de plan (Waterfall) sau dezvoltarea agila: https://www.seguetech.com/waterfall-vs-agile-methodology/

Metodologii agile

- Extreme programming
- Scrum
- Crystal
- Dynamic Systems Development Method (DSDM)
- Feature Driven Development (FDD)
- Kanban, s.a.

Extreme programming (XP)

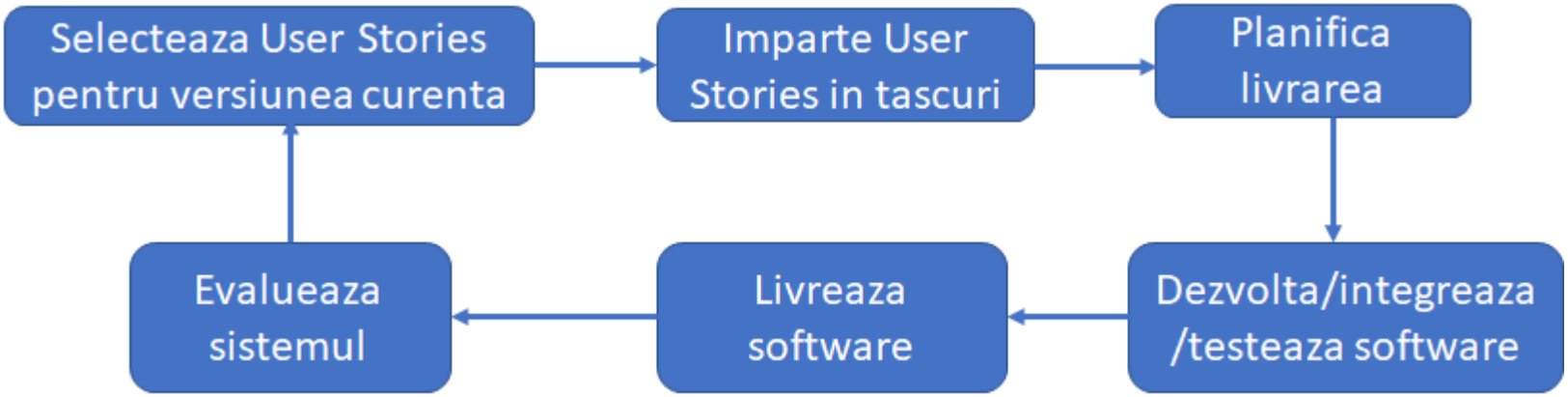
- Aparuta la sfarsitul anilor ’90
- **Abordare “extrema” in dezvoltarea iterativa**
  - Versiuni noi pot fi construite de mai multe ori pe zi
  - Incrementele sunt livrate clientilor la intervale de 2 saptamani
  - Pentru fiecare versiune executabila (build) trebuie rulate toate testele si versiunea este acceptata numai daca testele au trecut cu succes.

Practici cheie

- “User Stories” pentru specificare
- Test driven development: Test first development + Refactorizare
- Pair programming

XP – User stories

- In XP, un reprezentant al utilizatorilor finali/clientului face parte din echipa de dezvoltare si este responsabil cu cerintele.
- Cerintele utilizator sunt exprimate sub forma de scenarii, numite “user stories”.
- User story trebuie sa fie o descriere scurta, fiind limitata la o pagina de hartie de dimensiuni mici (de obicei 3x5 inches), impiedicand astfel cresterea descrierii.
- Reprezinta o modalitate simpla si rapida de a capta cerintele clientului.
- Pornind de la scenarii dezvoltatorii definesc tascuri de implementat.
- Pe baza tascurilor se fac estimarile de cost si timp.
- Clientul alege cerintele de implementat in urmatorul livrabil pe baza prioritatii lor si a estimarilor de timp.



User story

Kate is a doctor who wishes to prescribe medication for a patient attending a clinic. The patient record is already displayed on her computer so she clicks on the medication field and can select current medication', 'new medication' or 'formulary'.

If she selects 'current medication', the system asks her to check the dose. If she wants to change the dose, she enters the dose and then confirms the prescription.

If she chooses 'new medication', the system assumes that she knows which medication to prescribe. She types the first few letters of the drug name. The system displays a list of possible drugs starting with these letters. She chooses the required medication and the system responds by asking her to check that the medication selected is correct. She enters the dose and then confirms the prescription.

If she chooses 'formulary', the system displays a search box for the approved formulary. She can then search for the drug required. She selects a drug and is asked to check that the medication is correct. She enters the dose and then confirms the prescription.

The system always checks that the dose is within the approved range. If it isn't, Kate is asked to change the dose.

After Kate has confirmed the prescription, it will be displayed for checking. She either clicks 'OK' or 'Change'. If she clicks 'OK', the prescription is recorded on the audit database. If she clicks on 'Change', she reenters the 'Prescribing medication' process.

http://iansommerville.com/software-engineering-book/

XP - Test driven development (1)

Test driven development: Test first development + Refactorizare

- **Test-first development**: programatorul scrie testele de nivel coborat inainte de a scrie codul
- Testele sunt dezvoltate incremental pornind de la scenarii
- Reprezentantului utilizatorului/clientului este implicat in dezvoltarea si validarea testelor
- Sunt folosite instrumente de testare automata
- Atunci cand este adaugata o functionalitate sunt rulate toate testele anterioare si cele noi – se verifica astfel ca noua functionalitate nu a introdus erori.

Refactorizare (imbunatatirea codului)

- Rescrierea codului, fara a modifica functionalitatea sa externa, pentru imbunatatirea unor atribute ale codului, cum ar fi: usurinta de intelegere, complexitatea, usurinta de intretinere
- Codul este mai usor de inteles, reduce necesitatea documentatiei
- Modificarile se fac mai usor deoarece codul este bine structurat si clar

XP - Test driven development (2)

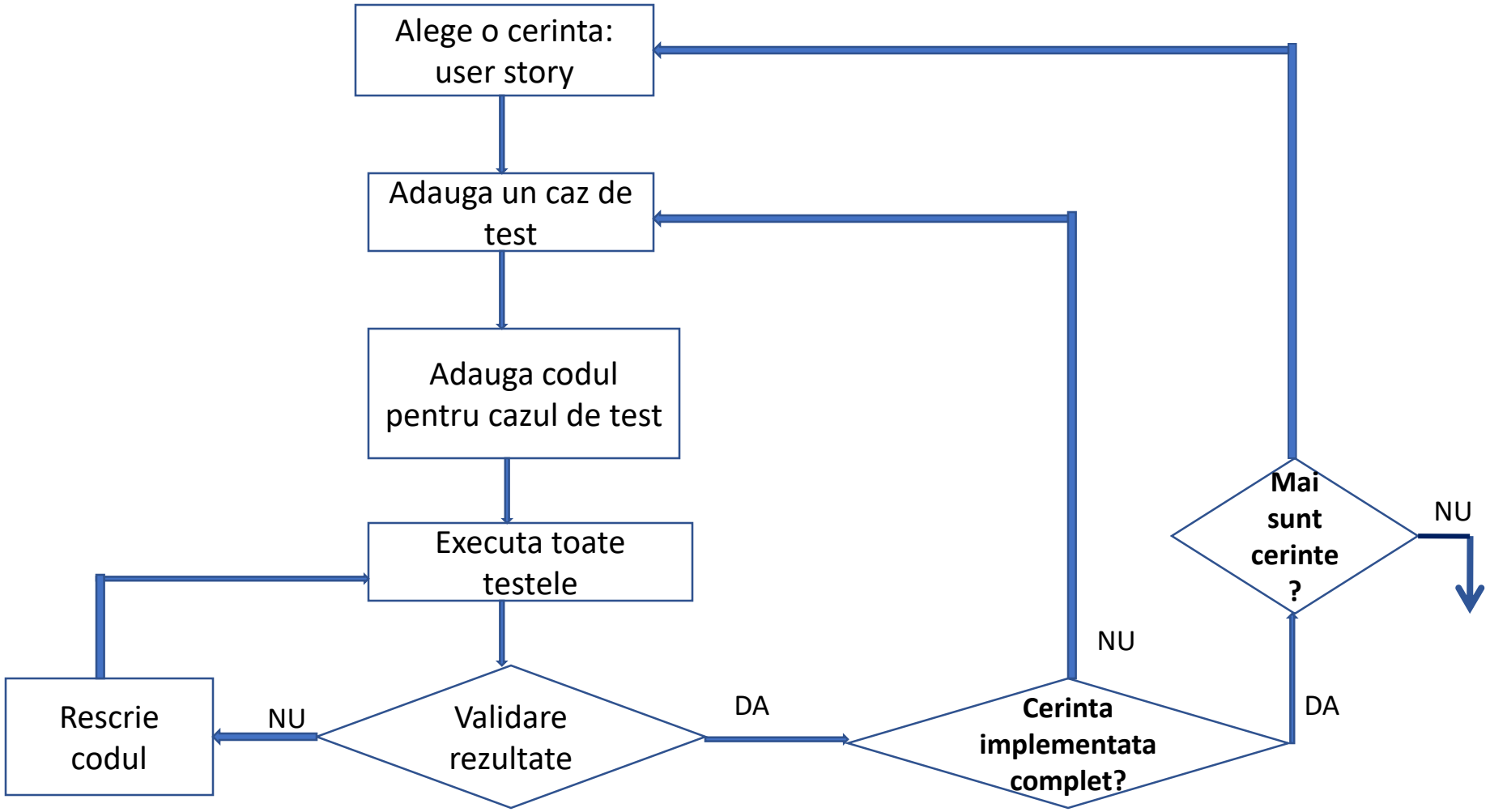
Procesul de testare si codificare in XP poate fi descris astfel:

Repeta cat timp mai exista cerinte neimplementate:

1. Se alege o cerinta functionala exprimata printr-o descriere a utilizatorului (user story)
2. Se scrie un caz de test care va verifica o parte a cerintei.
3. Se scrie codul care implementeaza partea din cerinta verificata prin cazul de test.
4. Se executa toate cazurile de test.
5. Daca rezultatele sunt validate, se reia procesul din pasul 2, pana la implementarea completa a cerintei,

   altfel, se rescrie codul si se revine in pasul 4.

XP - Test driven development (3)



XP – Pair programming

- Dezvoltatorii lucreaza in perechi, dezvoltad codul impreuna
- Este o revizie informala a codului – fiecare linie de cod este vazuta de 2 persoane
- Perechile sunt create dinamic, astfel incat toti membrii echipei lucreaza impreuna pe parcursul dezvoltarii

  → ajuta la cunoasterea codului de toata echipa

  → toti dezvoltatorii sunt responsabili pentru intregul cod (proprietate colectiva)
- Incurajeaza refactorizarea
- Dezvoltarea in echipa reduce riscul proiectului atunci cand un membru paraseste echipa.
- Dezvoltarea in perechi nu este ineficienta: exista dovezi care sugereaza ca este mai eficienta daca cei 2 ar lucra separat.

XP si principiile agile

- Dezvoltarea incrementala: incremente mici livrate frecvent
- Implicarea clientului: angajarea permanenta a clientul in echipa.
- Oameni nu procese: pair programming, proprietate colectiva, evitarea programului de lucru prelungit.
- Schimbarile acceptate prin livrari regulate.
- Mentinerea simplitatii prin refactorizarea continua a codului.

XP - limitari

- Fara teste/proceduri de acceptare specificate de client “user stories” sunt supuse interpretarii, ceea ce le face dificil de utilizat ca baza contractuala.
- Solicita contactul permanent cu clientul pe tot parcursul proiectului, dificil in unele cazuri, sau timp suplimentar consumat.
- Dificil de scalat la proiecte mari.
- Presupune dezvoltatori competenti.
- XP este greu de integrat cu practicile de management din cele mai multe companii
- Desi in dezvoltarea agila se folosesc practici din XP, metoda asa cum a fost definita initial nu este larg folosita.