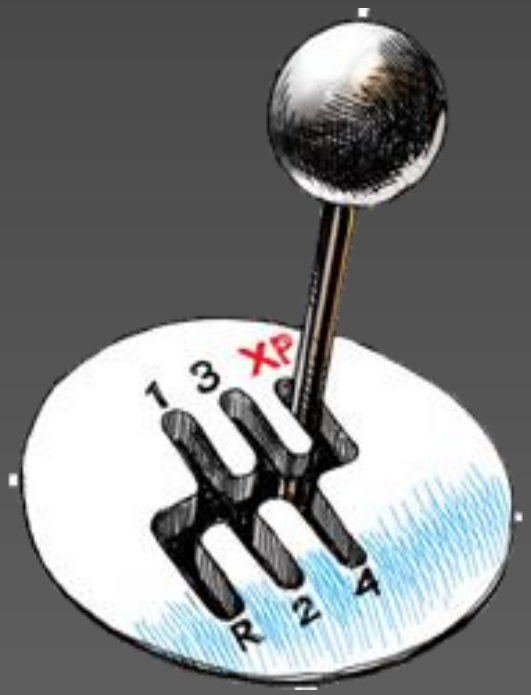
Extrém programozás

Konkoly Krisztián

Bevezető

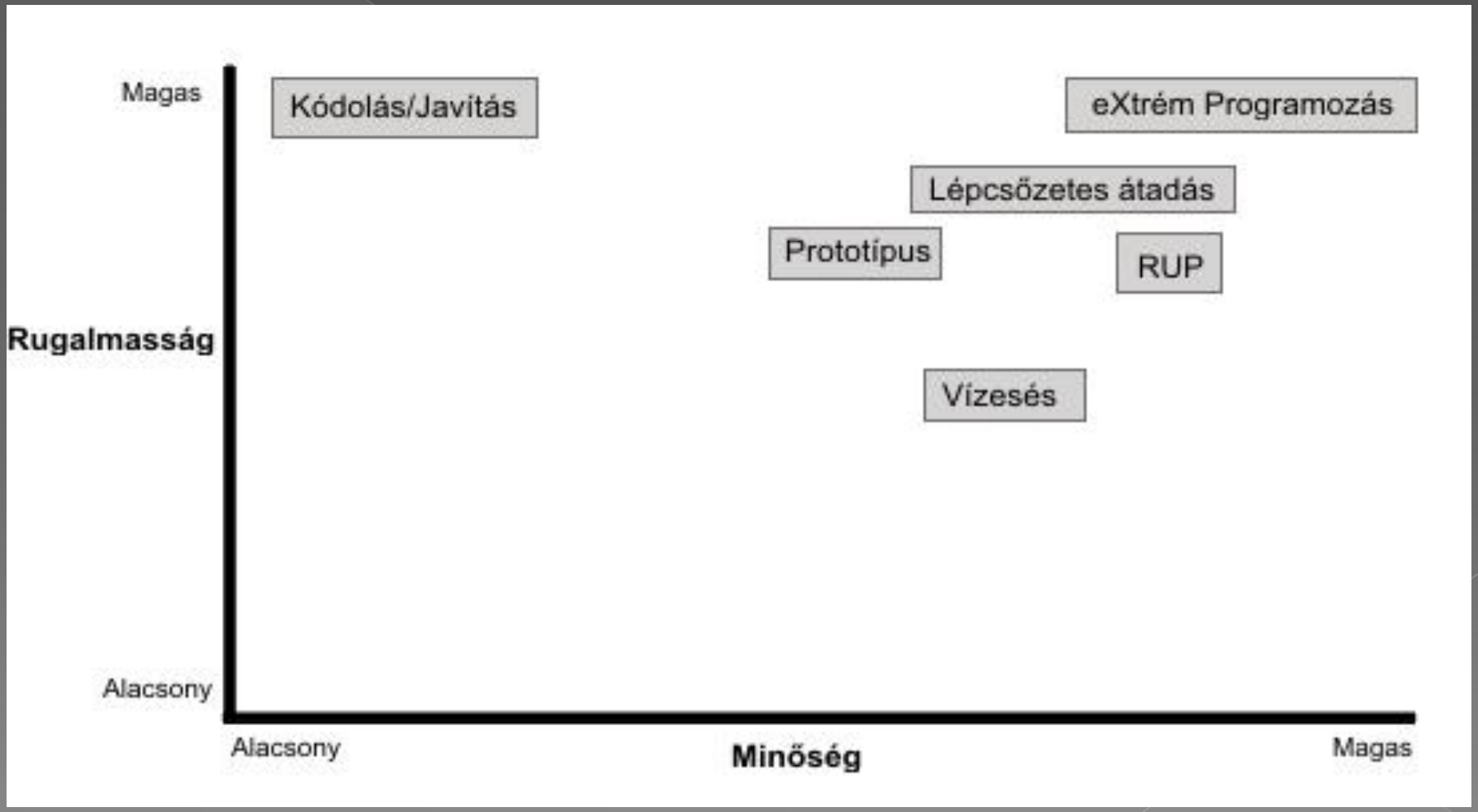


- Angolul: Extreme Programming, röviden: XP
- Agilis módszertan.
- Más módszertanok bevált technikáinak extrém módú (nagyon jó) használata

Agilis módszerek

- jelentése: gyors, fürge
- 1990-es évek vége
- Változás igénye
- Módszertan-család
- Emberhez közelebb áll

Módszertanok összehasonlítása



Az XP 4 tevékenységet ír elő

- Kódolás
  - Fontos elem
  - Problémák itt jelennek meg ténylegesen
  - Programozók közötti kommunikáció
- Tesztelés
  - XP szerint:
    Kevés teszt- kevés hiba
    Sok teszt- összes hiba
  - Tesztelés segíti a dokumentációt
  - Maga a teszt a dokumentáció
    (a metódusokat írja le)

Az XP 4 tevékenységet ír elő

- Odafigyelés
  - Kapcsolat a megrendelővel
  - Folyamatos kommunikáció
    (kivitelezhető-e?, Porojectek állapotai…)
- Tervezés
  - Fontos
  - Rugalmasság
    - esetleges későbbi változások miatt
    - Független komponensek
    - OOP elvek

Az XP 5 értéket vall

- Kommunikáció
- Egyszerűség
- Visszacsatolás
  - Visszacsatolás a rendszer felől
  - Visszacsatolás a megrendelő felől
  - Visszacsatolás a csapat felől (Planning Game)
- Bátorság
- Tisztelet

Néhány XP-re jellemző technika:

- Páros programozás (pair programming)
- Teszt vezérelt fejlesztés (test driven development)
- Forráskód átnézés (code review)

Néhány XP-re jellemző technika:

- Folyamatos integráció (continuous integration)
- Kódszépítés (refactoring)

Tesztelés XP-ben

- Nagyobb hangsúly itt a tesztelésen van
  - Fontos a rendszertesztelés!
  - A tesztelések jellemzői:
    - Előrehozott teszteléssel történő fejlesztés
    - Inkrementális tesztfejlesztés, forgatókönyv alapján
    - Felhasználók bevonása a tesztek fejlesztésébe
    - Automatizált tesztelő eszközök használata

Tesztelés XP-ben

- Az előrehozott teszteléssel történő fejlesztést az XP-nek köszönhetjük.
  - Meghatározzák előre az interfacet és a viselkedési specifikációkat.
  - A lényege hogy előbb írunk egy futó komponenst mint a kért feladatot
  - Egy későbbi dián részletezésre kerül jobban ez a téma.

Tesztelés XP-ben

- Elfogadási tesztelés: az ügyfél valós adataival történik a teszt.
- A teszt íróinak, tisztában kell lenni az program egészével(”teszt>implementáció”)
- Felléphetnek problémák
  - A programozók inkább fejlesztik az alkalmazást
  - A tesztelés olykor nehezebb mint a feladat leprogramozása

A Test-First development előnyei

- A tesztek csökkentik a bugok számát az új funkciókban.
  - Ha valamit elgépelünk, a tesztek nagy eséllyel kimutatják azt (hibás lesz néhány teszt)
- A tesztek jó dokumentációk.
  - A programozók számára érthetőbb egy jól strukturált program kód mint egy tömör dokumentáció.

A Test-First development előnyei

- A tesztek korlátozzák az osztályok feladatait.
  - Annyi kód, és programozásra van szükség amennyi a tesztekhez is kell. (Törekedjünk az egyszerűségre)
- A tesztek köszönhetően a programkód minősége javul.
  - Az elejétől úgy programozunk, hogy tesztelhető is legyen, így igényes jól tagolt, értelmezhető kódsorokat készítünk.

A Test-First development előnyei

- A tesztek megvédenek a bugok újra bevezetésétől.
  - Ha bug keletkezik, írunk rá egy tesztet, ami kiváltja, majd javítjuk a hibát. Később ha felmerül ez a hiba a teszt alapján ismerni fogjuk
- Felgyorsul a fejlesztési folyamat.
  - A tesztek írása maga lassú folyamat, de az egész fejlesztés felgyorsul, mivel például hibákat sem kell keresni

A Test-First development előnyei

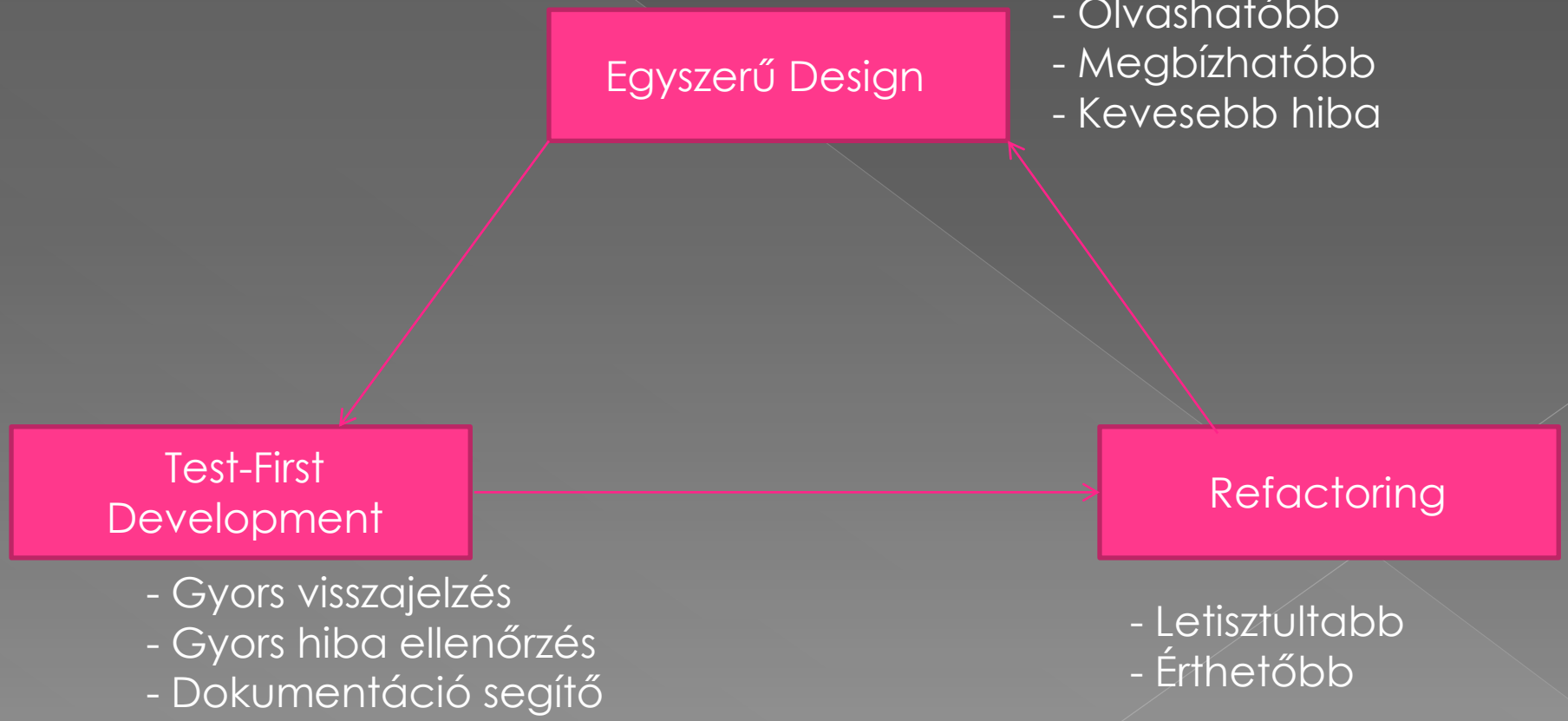
- A tesztek csökkentik a félreértelmezést.
  - Ha egy új funkciót vezetünk be, egy bughoz vezethet, ami lehet később jelenik meg, aminek a megtalálása hosszú időkbe telik. A kész tesztek viszont bebiztosítanak minket.

Test drive developement

- Test-First a teszt hibamentes futását követeli
- A Test-Driven az egyszerűsítést követeli meg
- Csak Test-First alkalamzása esetén bonyolult programot is kaphatunk
- Ez elkerülésére a Refactoringot alkalmazzuk!

Test drive developement

- A folyamat önfennartó



Test drive developement

- Az új funkciók könnyebb teszteléséhez és implementálásához fontos az egyszerű designe
- Ennek megvalósításában a refactoring segít
- Ez magával vonzza az automatizált teszteknek köszönhető ”védőhálót”
- És még több előnyök(Pl.: tiszta kód, öndokumentáció…)

Problémák

- Főként az ügyfél miatt lehetnek problémák
  - Folyamatosan változtathatja az elvárásait
  - Esetleg egyre többet akar
  - Ha jelen van a képviselő, sokakat zavarhat
  - Ha rosszul végzi a képviselő a munkát hatással van a projectre
- A követelmények nincsenek előre lepapírozva, mert az XP elvek szerint ez nem elég rugalmas.

Kérdések?

Köszönöm a Figyelmet!