Dr. Mileff Péter

Szoftverfejlesztés

Gyors szoftverfejlesztés,
Extreme Programming

1

TESZTELÉS AZ XP-BEN…

2

Tesztelés az XP-ben

- Az XP nagyobb hangsúlyt fektet a tesztelés folyamatára, mint a többi agilis módszer
  - A teszteléssel és a rendszer validálásával kapcsolatos problémák elkerülése miatt.
  - A rendszertesztelés központi szerepet játszik az XP-ben
    - Egy olyan megközelítést dolgoztak ki:
      - ami csökkenti annak a valószínűségét, hogy az új rendszerinkremensekkel hibák kerüljenek a már létező szoftverbe.

3

Tesztelés az XP-ben

- A tesztelés kulcsjellemzői:
  - 1. Előrehozott teszteléssel történő fejlesztés
  - 2. Inkrementális tesztfejlesztés a forgatókönyvek alapján
  - 3. Felhasználók bevonása a tesztek fejlesztésébe és validálásába
  - 4. Automatizált tesztelő eszközök használata.

4

Tesztelés az XP-ben

- Az XP egyik legfontosabb újítása az **előrehozott teszteléssel történő fejlesztés**
  - **(Test-First Development)**.
- Az előre elkészített tesztek:
  - mind az interfészt, mind a viselkedési specifikációt meghatározza a fejlesztendő funkcionalitás számára.
  - Ez csökkenti a követelményproblémák és interfészek félreértéseit.
- Ez a folyamat minden olyan folyamat esetén alkalmazható,
  - ahol tisztán látható a rendszerkövetelmény és az implementálandó kód közötti kapcsolat.

5

Tesztelés az XP-ben

- Az extrém programozásban mindig látható ez a kapcsolat
  - mert a követelményeket reprezentáló történetkártyák feladatokra vannak bontva,
  - amelyek egy implementációs egységet képeznek.
- Minden egyes feladathoz egy vagy több különálló egységteszt tervezhető
- Az ügyfél szerepe a tesztelési folyamatban:
  - a történetekhez segítsen elfogadási teszteket fejleszteni,
    - amelyeket a rendszer következő kiadásában kell implementálni.

6

Tesztelés az XP-ben

- **Elfogadási tesztelés:** az a folyamat, amikor a rendszert az ügyfél adataival ellenőrizzük,
  - hogy megfelel-e az ügyfél valódi elvárásainak.
- Az elfogadási tesztelés ugyanúgy inkrementális, mint a fejlesztés.
- A gyakorlatban az elfogadási tesztek egész sorát szokták készíteni.
- Az előrehozott teszteléssel történő fejlesztés és az automatikus teszteszközök használata az XP-irányzat fontos erősségei.

7

Tesztelés az XP-ben

- Az előrehozott tesztelés jelentése:
  - a tesztelést, mint futtatható komponens hamarabb megírjuk, mint a feladatot.
  - Így amint kész a szoftver implementációja, rögtön futtatható rajta a teszt.
  - A tesztelő komponensnek függetlennek kell lennie,
    - azaz szimulálnia kell a tesztelendő bemenetek megadását,
    - és ellenőriznie kell, hogy a végeredmény megfelel-e a kimenet specifikációjának.
  - Az automatikus teszteszköz olyan eszköz, amely elküldi ezeket az automatikus teszteket végrehajtásra.

8

Tesztelés az XP-ben

- Az előrehozott teszteléssel történő fejlesztés esetében:
  - a feladat implementálóinak átfogóan kell érteniük a specifikációt,
  - hogy teszteket készítsenek a rendszerhez.
- Ez azt jelenti:
  - a specifikációban lévő félreérthetőségeket és hiányosságokat még az implementáció megkezdése előtt tisztázni kell.
- A módszer kikerüli a „*lemaradt teszt*” problémáját

9

Tesztelés az XP-ben

- Mikor lép fel?
  - amikor a rendszer fejlesztője nagyobb léptékben dolgozik, mint a tesztelő,
  - így az implementáció egyre jobban megelőzi a tesztelést.
  - Így néha kihagyhatnak teszteket, hogy tartani tudják az ütemtervet.
- Az előrehozott teszteléssel történő fejlesztés nem mindig működik úgy, ahogy várnánk:
  - a programozók jobban szeretnek programozni, mint tesztelni
  - sokszor hiányos teszteket írnak.

10

A TEST-FIRST DEVELOPMENT ELŐNYEI…

11

A Test-First development előnyei

- **A tesztek csökkentik a bug-ok számát az új funkciókban.**
  - Mivel a teszteket hamarabb írjuk, mint a tényleges programkódot,
    - egy esetleges elgépelés nagy valószínűséggel el fog rontani néhány tesztet.
- **A tesztek jó dokumentációk.**
  - Megfigyelések bizonyítják:
    - egy programozó sokkal hamarabb megérti egy programegység funkcióját egy szemléletes programkódból,
      - mint folyamatos szövegben írt dokumentáció alapján.

12

A Test-First development előnyei

- **A tesztek korlátozzák az osztályok feladatait.**
  - A kódolás során csak annyit kódolunk, amennyi a tesztek kielégítéséhez szükséges.
  - Ennél fogva ami nem szerepel a tesztben, annak nem kell megjelennie a programkódban sem.
  - Nem szükséges keretrendszereket és a későbbi bővítés számára fenntartott csatlakozási pontokat beiktatni a kódba.
  - Törekedjünk inkább egyszerű design-ra, ami minden feladatot ellát.

13

A Test-First development előnyei

- **A tesztek javítják a programkód minőségét.**
  - A TFD miatt már kezdettől fogva úgy tervezzük a programunkat, hogy az tesztelhető legyen.
  - Ezt csak jól tagolt, moduláris kóddal érhetjük el, aminek további hasznos következményei vannak.
- **A tesztek megvédnek a bug-ok újra bevezetésétől.**
  - Ha a programban bug-ot találunk, akkor először megismételtetjük az egy új, eddig a tesztgyűjteményben még nem szereplő teszttel,
  - majd kijavítjuk a hibát.
  - Ha valaki módosítja a kódunkat, és újra bevezeti ezt a bug-ot, akkor a teszt azonnal jelezni fog.

14

A Test-First development előnyei

- **Felgyorsul a fejlesztés**
  - A kódolást tekintve a tesztkészítés lelassítja a programozót:
    - a tesztesetek megfogalmazása, elkészítése időbe telik.
  - Az egész fejlesztés sebessége viszont felgyorsul, mert kevesebb időt töltünk hibakereséssel,
  - és az új funkciók implementálása is gyorsabban megy,
    - mert nem kell állandóan azon aggódni, hogy elrontjuk a már kész kódot.

15

A Test-First development előnyei

- **A tesztek csökkentik a félelemérzetet**
  - A programozók legnagyobb rémálma az, amikor egy jól működő funkcióba vezetünk be bug-ot,
    - Amely csak hetek múlva jelenik meg.
  - A jó minőségű tesztgyűjtemény mint egy „védőháló" vigyáz a programozóra,
    - és az esetek 90%-ában jelez, ha megbontottunk egy már jó funkciót.
    - A programozó így nem fog habozni, ha egy jól tesztelt kódot kell módosítania.

16

TEST-DRIVEN DEVELOPMENT…

Test-Driven Development

- A Test-First fejlesztés csak azt követeli meg:
  - hogy minden teszt sikeresen fusson le.
- A Test-Driven fejlesztés viszont:
  - megköveteli, hogy minden funkció implementálása után gondoljuk át, lehet-e egyszerűsíteni a rendszeren.
  - Ha erre a kérdésre igen a válasz, akkor a refactoring technikájával ezt oldjuk is meg.

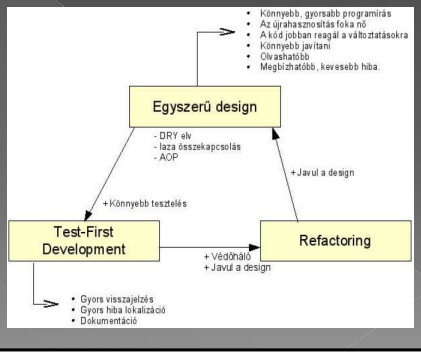
Test-Driven Development

- Ha szigorúan csak a TFD elveihez ragaszkodnánk:
  - a programunk a funkciók bővítése során egyre kaotikusabbá és bonyolultabbá válna,
    - mivel semmi mással nem törődünk, csak azzal, hogy a teszteket kielégítsük.
- A refactoring lépés pontosan ezt akadályozza meg,
  - így folytonosan be tudjuk tartani „egyszerű design" elvét.

Test-Driven Development

- Ez folyamat egy pozitív visszacsatolási folyamat
  - azaz amellett hogy a rendszer önfenntartó, további artifact-öket termel.
  - Az egyes lépések támogatják a soron következőt,
    - ez az, ami a Test Driven Development jelentőségét adja.

Test-Driven Development



Test-Driven Development

Egy pozitív visszacsatolási rendszer:

- 1. Az új funkciók egyszerű teszteléséhez és implementálásához elengedhetetlen az egyszerű design.
- 2. Az egyszerűség eléréséhez a refactoring-ot alkalmazzuk.
- 3. A refactoring viszont elképzelhetetlen az automatizált tesztek nyújtotta „védőháló" nélkül.
- 4. Ezen túl a TDD komponenseinek további, a szoftverfejlesztő számára fontos haszna is van.
  - (Pl. a gyors visszajelzés, öndokumentáló, tiszta kód, stb.)

VERIFIKÁCIÓ ÉS VALIDÁCIÓ...

Verifikáció és validáció

- Az implementációs folyamat közben és után az éppen fejlesztett programot ellenőrizni kell
  - megfelel-e megfelel a specifikációnak és kielégíti a megrendelő igényeit.
- Az ellenőrzőfolyamatok neve ***verifikáció*** és ***validáció***.
  - Ezek a folyamatok a fejlesztés minden lépésében jelen vannak.
  - **Validáció:** A megfelelő terméket készítetjük el?
  - **Verifikáció:** a terméket jól készítetjük el?

Verifikáció és validáció

- A **verifikáció**: magába foglalja annak ellenőrzését, hogy a szoftver megfelel-e a specifikációnak,
  - azaz eleget tesz-e a funkcionális és nem funkcionális követelményeknek.
  - Általánosabban: a szoftver megfelel-e a vásárló elvárásainak.
- A **validáció**: ennél kicsit általánosabb fogalom.
  - Végcélja:
    - megbizonyosodjunk arról, hogy a szoftverrendszer „megfelel-e a célnak".
    - Azaz teljesül-e a vásárló elvárása,
      - amibe beleértendő olyan nem funkcionális tulajdonságok is, mint a hatékonyság, hibatűrés, erőforrásigény.

25

Verifikáció és validáció

- A V & V folyamaton belül a rendszer ellenőrzésére két egymást kiegészítő technika használható:
  - **Statikus:** szoftverátvizsgálások, amelyek a rendszer reprezentációját elemzik:
    - Követelmény dokumentumot, terveket, és forráskódot.
  - **Dinamikus:** szoftvertesztelés, amely csakis az implementáció fázisában végezhető el.
    - A tesztadatok segítségével ellenőrzi, hogy az megfelelő teljesítményt nyújt-e.

26

Verifikáció és validáció

- Az átvizsgálási folyamat a szoftverfolyamat bármely lépésében használhatók.
- A követelményekkel kezdődően minden olvasható reprezentáció átvizsgálható.
- Az átvizsgálási technikák:
  - a *programátvizsgálások, az automatizált forráskódelemzés és formális verifikáció.*
- A rendszert csak akkor tesztelhetünk ha elkészült egy végrehajtható változatának prototípusa.

27

Verifikáció és validáció

- Az inkrementális fejlesztés előnye:
  - a különböző inkremensek külön tesztelhetők, és az egyes funkciók már hamarabb tesztelhetők.
- A szoftvertesztelés, mint dinamikus V & V a gyakorlatban inkább alkalmazott technika.
- Ekkor a programot a valós adatokhoz hasonló adatokkal teszik próbára.
  - A kimenetek eredményei lehetőséget adnak anomáliák, problémák feltárására.

28

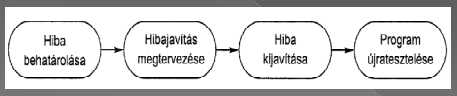
Verifikáció és validáció

- Ezen tesztelés két fajtája ismert:
  - **Hiányosságtesztelés:** célja a program és a specifikációja között meglévő ellentmondások felderítése.
  - Az ilyen teszteket a rendszer hiányosságainak feltárására tervezik, nem a valós működés szimulálására.
  - **Statisztikai (vagy validációs) tesztelés:** a program teljesítményének és megbízhatóságának tesztelése valós körülményeket szimulálva.
  - Annak megmutatása, hogy a szoftver megfelel-e a vásárlói igényeknek.

29

Verifikáció és validáció

- A V & V folyamatokat gyakran a belövési folyamattal ötvözik.
- **A belövés az a folyamat, amely behatárolja és kijavítja a szoftverben talált hiányosságokat.**



A belövésre nem létezik egyszerű módszer, mert a hibák behatárolása a programban nem mindig könnyű

30

Verifikáció és validáció

- A hiba nem mindig ott található meg, ahol a sikertelenség bekövetkezett.
- Bizonyos esetekben szükség lehet új tesztek megtervezésére,
  - amelyek megismétlik az eredeti hibát és interaktív belövő eszközök (pl. Debugger) használatára.
- Bármely a rendszerben talál hiba után az összes tesztet meg kell ismételni,
  - azonban a gyakorlatban ezt nem alkalmazzák, mert nagyon költséges.

31

VERIFIKÁCIÓ- ÉS VALIDÁCIÓTERVEZÉS...

32

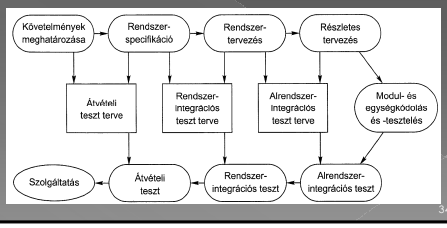
Verifikáció- és validációtervezés

- A V & V költséges folyamat,
  - bonyolult valós idejű rendszereknél akár a költségek 50%-át is elérheti.
  - Ezért gondosan meg kell tervezni.
- **Egy szoftverrendszer verifikációjának és validációjának tervezését még a fejlesztési folyamat elején el kell kezdeni.**

33

Verifikáció- és validációtervezés

- **V modell:**
  - megmutatja, hogy hogyan kapcsolják össze a teszttervek a tesztelési és fejlesztési tevékenységeket.



34

Köszönöm a figyelmet!

35