

Dr. Mileff Péter

Szoftverfejlesztés

V & V, Szoftvertesztelés

Miskolci Egyetem
Általános Informatikai Tanszék

1

STATIKUS TECHNIKÁK A V & V-BEN...

A V & V tervezési folyamatoknak egyensúlyt kell kialakítani a verifikáció és a validáció statikus és dinamikus technikái között.

2

Statikus technikák: A szoftver átvizsgálása

- A szisztematikus programtesztelés időigényes és drága folyamat.
- Minden tesztfuttatás egyetlen vagy legfeljebb néhány hibát derít fel.
- Ezzel ellentétben a **programkód átvizsgálása** hatékonyabb és olcsóbb megoldás.
- A program hibáinak több, mint 60%-a felderíthető programátvizsgálással.

3

Statikus technikák: A szoftver átvizsgálása

- A programkód átvizsgálásának hatékonyságát két ok magyarázza:
 - Egy menetben több, független hiba is kiderülhet.
 - Az átvizsgálók felhasználják a szakterületre és a programozási nyelvre vonatkozó ismereteiket.
 - Valószínűleg találkoztak már régebben is ilyen hibafajtákkal, így tudják, mire kell figyelni.
- A programkód átvizsgálását egy legalább 4 emberből álló csapatnak érdemes végeznie.

4

Statikus technikák: A szoftver átvizsgálása

- A szerző, az olvasó, a tesztelő és a moderátor.
- A csapat tagjai szisztematikusan elemzik a kódot és rámutatnak az esetleges hibákra.
- Maga az átvizsgálás max. 2 óra, amely során az olvasó felolvassa a kódot.
- Végül a szerző módosítja a programot.
- Ezután vagy újra átvizsgálják a kódot, vagy a moderátor úgy dönt, hogy ez nem szükséges.
 - Az átvizsgálás folyamatát a szokásos programozási hibák – nyelvtől függő – listájára kell alapozni.

5

Statikus technikák: A szoftver átvizsgálása

- A szoftver átvizsgálás eredményes folyamat
- Mi a probléma?
 - Ennek ellenére sok szoftverfejlesztő cégnél nehéz bevezetni.
- Oka:
 - a tesztelésben jártas szoftvertervezők vonakodva fogadják el a módszer hatékonyságát.

6

Statikus technikák: Automatizált statikus elemzés

- A statikus programelemzők:
 - olyan szoftvereszközök, amelyek a program forrásszövegének vizsgálatával derítik fel a lehetséges hibákat és anomáliákat.
 - Ehhez nincs szükség a program futtatására.
 - Észlelhetik:
 - az utasítások formailag helyesek-e, a nem használt kódrészleteket, inicializálatlan változókat, stb.
 - Kiegészítik a nyelv fordítóprogramja által adott lehetőségeket.

7

Statikus technikák: Automatizált statikus elemzés

- Néhány statikus elemzéssel felismerhető hibafajta és anomália:
 - (az anomália nem feltétlenül programhiba, lehet következmény nélküli is):
 - Adathibák,
 - Vezérlési hibák,
 - Input/output hibák,
 - Interfészhibák.

8

SZOFTVERTESZTELÉS...

9

Bevezetés

- A szoftvertesztelésnek tehát két célja van:
 - **1. Bizonyítani a fejlesztő és a vásárló számára, hogy a szoftver megfelel a vele szemben támasztott követelményeknek.**
- Egyedi szoftver esetén ez azt jelenti:
 - a felhasználói és rendszerkövetelmények leíró dokumentum minden követelményére vonatkozóan legalább egy tesztnek lennie kell.
 - Általános szoftver esetén pedig minden egyes rendszertulajdonságra kell lenni egy tesztnek.

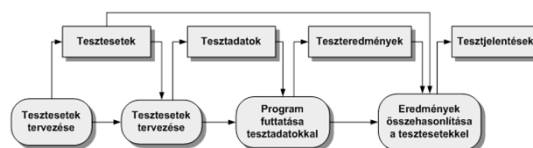
10

Bevezetés

- **2. Felfedezni a szoftverben azokat a hibákat és hiányosságokat, amelyekben a szoftver viselkedése helytelen, nemkívánatos, vagy nem felel meg a specifikációnak.**
- A hiányosságtesztben felfedett hibák kiirtásával foglalkozik.

11

A tesztelés folyamatának általános modellje



12

A tesztelés folyamatának általános modellje

- **A teszteset:**
 - nem más, mint a teszthez szükséges inputok és a rendszertől várt outputok specifikációja.
- A tesztadatok kifejezetten a rendszer tesztelésére létrehozott inputok.
- Néha automatikusan generálhatók,
 - az automatikus teszteset-generálás viszont lehetetlen.
- A tesztek outputjait csak azok tudják előre megjósolni, akik értik, hogy a rendszernek mit kellene csinálnia.

13

A tesztelés folyamatának általános modellje

- Beszélhetünk **kimerítő tesztelés**ről is.
 - ahol az összes lehetséges program-végrehajtási szekvenciát teszteljük.
 - A gyakorlatban nem praktikus,
 - ezért a tesztelésnek a lehetséges tesztesetek egy részhalmazán kell alapulnia.
 - Erre irányelveket kell kidolgozni a szervezetnek,
 - nem pedig a fejlesztőcsoportra hagyni.

14

RENDSZERTESZTELÉS...

15

Rendszertesztelés

- **A rendszertesztelés:**
 - két vagy több, rendszerfunkciót vagy jellemzőt megvalósító komponens integrálását és az integrált rendszer tesztelését jelenti.
- A rendszertesztelés iteratív fejlesztési folyamatban:
 - a vásárló számára leszállítandó inkremens,
 - míg vizesés jellegű folyamat során a teljes rendszer tesztelését jelenti.

16

Rendszertesztelés

- A legtöbb bonyolult rendszer esetében a rendszertesztelést két külön fázisra bonthatjuk:
 - **1. Integrációs tesztelés:**
 - ahol a tesztelést végző csapat hozzáfér a rendszer forráskódjához.
 - Leginkább a rendszer hiányosságainak megtalálásával foglalkozik.

17

Rendszertesztelés

- **2. Kiadástesztelés:** ahol a rendszer egy felhasználók számára kiadható verziója kerül tesztelésre.
 - Itt a tesztcsoport azt vizsgálja, hogy a rendszer megfelel-e a követelményeknek.
 - A kiadástesztelés általában fekete doboz tesztelés:
 - ahol a teszt csapatnak egész egyszerűen csak azt kell bemutatnia, hogy a rendszer jól működik-e, avagy sem.
 - Ha ebbe a kiadástesztelésbe a vásárlót is bevonják,
 - akkor ezt **elfogadási tesztelésnek** nevezzük.
 - Ha a verzió elég jó, a vásárló elfogadhatja használatra.

18

Rendszertesztelés

- Az integrációs tesztelésre alapvetően úgy kell tekinteni,
 - mint rendszerkomponensek egy csoportjából vagy fűrtjéből álló befejezetlen rendszerek tesztelésére.
- A kiadástesztelés a rendszer azon kiadásának tesztelésével foglalkozik, amelyet le szeretnénk szállítani a vásárlóknak.

19

INTERGRÁCIÓS TESZTELÉS...

20

Integrációs tesztelés

- **A rendszer-integráció folyamata magában foglalja:**
 - a rendszer komponenseiből történő felépítését
 - és az eredményül kapott rendszer tesztelését a komponensek együttműködéséből adódó problémák felderítésére.
- **Az integrációs tesztelés ellenőrzi:**
 - hogy ezek a komponensek valóban képesek-e együttműködni,
 - megfelelően vannak-e meghívva
 - és interfészeiken keresztül a megfelelő adatokat a megfelelő időpontban küldik-e át.

21

Integrációs tesztelés

- **A rendszer-integráció során:**
 - azonosítani kell a rendszer különböző funkcionalitásait biztosító komponensek csoportjait,
 - majd további kód hozzáadásával integrálni kell őket.
- Sok esetben elsőként a rendszer váza készül el, és ehhez adódnak hozzá a komponensek.
- Ezt **fentről lefelé** történő integrációnak nevezzük.

22

Integrációs tesztelés

- **Más esetekben:**
 - először a gyakran a közös szolgáltatásokat biztosító komponensek integrálását végezzük először
 - Pl.: a hálózati és az adatbázis elérés.
 - majd hozzáadjuk a funkcionális komponenseket.
- Ez a ***lentől felfelé*** történő integráció.
- A gyakorlatban sokszor ezek keverékét alkalmazzák.

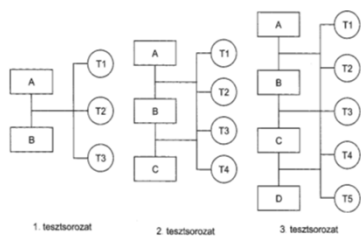
23

Integrációs tesztelés

- A rendszer integrálása és tesztelése során mindig inkrementális megközelítést kell alkalmazni.
- **Ennek oka:**
 - Az integráció folyamata során felmerülő hibák megtalálása nehéz feladat,
 - mert rendszerkomponensek között komplex együttműködés zajlik.

24

Inkrementális integrációs tesztelés



25

Integrációs tesztelés

- Az integráció megtervezése:
 - mindig döntenünk kell a komponensek integrációjának sorrendjéről.
- Olyan esetekben, amikor a vásárló nincs bevonva a folyamatba:
 - a leggyakrabban használt funkcionalitást megvalósító komponenseket célszerű elsőként integrálni,
 - azaz azokat, amelyeket a legtöbbet teszteltük.

26

Regressziós tesztelés

- **Regressziós tesztelés:** Egy meglévő teszt sorozat újbóli lefuttatását jelenti.
 - Akkor van rá szükség, ha egy új komponens integrálunk és tesztelünk.
 - Az új komponens integrálása megváltoztathatja a korábbi, már tesztelt komponensek közötti együttműködések mintáját.
 - Olyan hibákat is felfedezhetünk, amelyek az egyszerűbb konfiguráció tesztelésénél nem jelentkeztek.
 - Ezért nem szabad csakis az új komponensre vonatkozó teszteseteket futtatni, hanem a korábbiakat is.

27

Regressziós tesztelés

- Igen költséges folyamat,
- A gyakorlati alkalmazásához automatizált támogatásra van szükség.
 - Pl.: JUnit tesztelés.

28

Kiadásteresztek

- **Kiadásteresztesítés:** az a folyamat, amelynek során a rendszervásárlóknak leszállítandó kiadás (verzió) tesztesése történik.
- **A folyamat elsődleges célja:**
 - megmutassa, a rendszer megfelel a követelményeinek
 - funkcionalitás, teljesítmény, üzembiztonságra vonatkozóan.
- Egy **fekete doboz folyamat**, ahol a teszteseket a rendszer specifikációjából származtatják.

29

Kiadásteresztek

- **A rendszert fekete dobozként kell tekinteni:**
 - viselkedésére csak bemeneteinek és kimeneteknek a vizsgálatával lehet következtetni.
 - Más néven ezt *funkcionális tesztesítésnek* is nevezik.
- **A kiadások tesztesése során:**
 - meg kell próbálni „elrontani” a szoftvert
 - oly módon, hogy kiválasszuk azokat a bemeneteket, amelyek nagy valószínűséggel rendszerhibát okoznak.

30

Teljesítménytesztesítés (stresszteszt)

- Amikor teljesen integráltuk a rendszert,
 - lehetséges annak eredendő tulajdonságainak tesztesése,
 - mint pl. teljesítmény és a megbízhatóság tesztesése.
- **Miért készítünk teljesítménytesztesiteket?**
 - mert biztosítani szeretnénk, hogy a tervezett terhelés mellett a rendszer képes dolgozni.
- Általában olyan tesztesetek sorozata, ahol a terhelés addig nő, amíg a rendszer teljesítménye elfogadhatatlanná nem válik.

31

Teljesítménytesztesítés (stresszteszt)

- A rendszereket általában megadott terhelésre tervezik.
 - A stresszteszt a tervezett maximális terhelésen túl is folytatja a tesztesetet, mindaddig, amíg a rendszer hibázik.
- **Okai:**
 - **1. Teszteli a rendszer viselkedését szélsőséges körülmények között.**
 - Ilyen körülmények között fontos, hogy a túlterhelés ne okozzon adatvesztést vagy a felhasználói szolgáltatások váratlan eltűnését.

32

Teljesítménytesztelés (stresszteszt)

- **2. Terheli a rendszert, amellyel olyan hiányosságok is napvilágra kerülnek, amelyek normális körülmények között nem jelennek meg.**
 - Ezek a hiányosságok normál használat során nem okoznának rendszerhibát,
 - de felléphet a normál körülmények váratlan kombinációja, amit a stressztesztelés előidéz.

33

Teljesítménytesztelés (stresszteszt)

- A stressztesztelés különösen elosztott rendszereknél lényeges.
- Ezek a rendszerek gyakran nagy teljesítményromlást mutatnak, amikor nagyon leterheltek.
- A hálózat ilyenkor telítődik a koordinációs adatokkal,
 - amiket a folyamatok küldenek egymásnak, és mivel a többi folyamattól szükséges adatokat eközben várják.

34

Komponenstesztelés

- A komponenstesztelés (amit néha *egységtesztelésnek* is neveznek):
 - a rendszer önálló komponenseinek tesztelése.
 - Ez egy hiányosságtesztelési folyamat.
 - mivel célja a vizsgált komponensekben lévő hibák felderítése.
 - A legtöbb rendszerben a komponens teszteléséért annak fejlesztője felelős.
- A tesztelhető komponensek különbözők lehetnek.

35

Komponenstesztelés

- **Komponensek:**
 - 1. Önálló függvények vagy objektumok esetén módszerek.
 - 2. Több attribútumot és módszert tartalmazó objektumosztályok.
 - 3. Különböző objektumokból és/vagy függvényekből készült összetett komponensek,
 - amelyek funkcionalitását interfészekon keresztül érhetjük el.

36

2011.04.25.

Köszönöm a figyelmet!

37