Dr. Mileff Péter

Szoftverfejlesztés

Szoftvertesztelés, Kialakuló technológiák

Interfésztesztelés

- Az interfésztesztelésre mikor kerül sor?
  - amikor egy nagyobb rendszer létrehozásához modulokat és alrendszereket integrálunk,
    - amelyek egymással interfészeken keresztül kommunikálnak.
- Ez a fajta tesztelés különösen fontos az objektum-orientált és a komponens alapú programozásnál
  - ahol osztályokat, objektumokat és komponenseket újrafelhasználunk.
  - Itt a hibák inkább az objektumok között interakciók eredményei nem az egyedi objektumok viselkedéséből adódnak.

Interfésztesztelés

- A programkomponensek között különböző típusú interfészek léteznek.
  - következésképpen különböző típusú interfészhibák fordulhatnak elő.
  - **1. Paraméter-interfészek:**
    - Ezek olyan interfészek, ahol az adat- vagy néha a függvényreferenciák továbbítódnak az egyik komponenstől a másikhoz.
  - **2. Osztott memóriájú interfészek:**
    - Olyan interfész, ahol egy memóriablokk van megosztva az alrendszerek között.
    - Az adatokat az egyik alrendszer a memóriába írja, ahonnan egy másik alrendszer kiolvassa.

Interfésztesztelés

- **3. Procedurális interfészek:**
  - Ezek olyan interfészek, ahol az egyik alrendszer a más alrendszerek által hívható eljárások egy halmazát bezárja.
- **4. Üzenettovábbító interfészek:**
  - Ezek olyan interfészek, ahol egy alrendszer valamilyen szolgáltatást kér egy másik alrendszertől úgy, hogy üzenetet továbbít hozzá.
    - A szolgáltatás lefuttatásával kapott eredményeket pedig egy válaszüzenet tartalmazza.

Interfésztesztelés

- Komplex rendszereknél a gyakori interfészhibák:
  - **Interfész téves alkalmazása:** főleg a paraméter interfészeknél fordul elő.
    - Rossz típusú, rossz sorrendű, nem megfelelő számú paraméter átadás.
  - **Interfész félreértelmezése:** A hívó komponens félreértelmezi a hívott specifikációját.
    - pl. a bináris keresést meghívjuk egy rendezetlen tömbbel, így a keresés hibás lesz.
  - **Időzítési hibák:** valós idejű rendszereknél fordul elő, amelyek osztott memóriájú vagy üzenettovábbító interfészt használnak.
    - Az adat előállítója és feldolgozója eltérő sebességgel üzemelhet, így a feldolgozó idejétmúlt adatot kaphat.

5

Interfésztesztelés

- Az interfészhibák tesztelése nehéz.
  - mert néhányuk csak szokatlan feltételek között jelentkezik.
  - Pl.: egy objektum fix hosszúságú struktúraként implementál egy sort.
  - A hívó objektum feltételezheti, hogy a sor végtelen adatstruktúraként lett megvalósítva, és nem ellenőrzi a túlcsordulást.
  - Ez a hiba csak akkor derül ki a tesztelés során, ha a teszteset kierőlteti a túlcsordulást.

6

Teszteset tervezés

- **Tesztesettervezés:**
  - a rendszer- és komponenstesztelés azon része, amikor a rendszer tesztelését végző tesztesetek tervezése történik.
  - A folyamat célja a program hiányosságainak hatékony felderítésére alkalmas tesztesetek kialakítása.
- A tesztesetek tervezésekor különböző lehetőségek közül választhatunk:
  - **Követelmény alapú tesztelés**
  - **Partíciós tesztelés**
  - **A strukturális tesztelés**

7

Teszteset tervezés

- **1. Követelmény alapú tesztelés:**
  - Amikor a teszteseteket a rendszer követelményeinek tesztelése céljából készítjük.
- **2. Partíciós tesztelés:**
  - Input és output partíciókat hozunk létre,
  - úgy tervezzük meg a teszteket, hogy a rendszer minden partícióból lefuttatja az inputokat,
  - és minden partícióba generál outputot.
- **3. A strukturális tesztelés:**
  - A program részeit vizsgáló teszteket a program szerkezetének ismeretében készítjük el.

8

Teszteset tervezés

- A tesztesetek tervezésének egyik alapelve:
  - a követelményekből kiindulva a legmagasabb szintű tesztekkel kezdjünk,
  - majd a részletesebb tesztekkel folyamatosan, a partíciós és a strukturális tesztek segítségével bővítsünk.

9

Követelményalapú tesztelés

- A követelmény-alapú tesztelés olyan szisztematikus teszteset tervezést jelent, amikor az egyes követelményekből tesztsorozatokat származtat.
- Ez a tesztelési mód inkább validáció, mintsem hiányosságtesztelés
- azt próbálja bemutatni, hogy a rendszer megfelelően megvalósítja a követelményekben leírtakat.

10

Partíciós tesztelés

- Egy program inputjai általában különböző osztályokba esnek.
- Ezek rendelkeznek valamilyen közös jellemzővel,
  - pl. pozitív vagy negatív számok, szóköz nélküli sztringek, stb.
- A programok általában az osztály minden tagjára hasonló módon viselkednek.
- Ezért nevezik ezeket ekvivalencia – osztályoknak vagy tartományoknak.

11

Partíciós tesztelés

- Ekvivalencia-osztály például az érvénytelen és az érvényes inputok halmaza is.
- A hiányosságtesztelés szisztematikus megközelítése az ekvivalencia-osztályok azonosításán alapul.
- Először meg kell határozni az osztályokat,
  - majd minden osztályból teszteseteket kell választani.
  - Az osztály határáról és közepéről is érdemes teszteseteket választani.
  - Az ekvivalencia-osztályok a programspecifikáció vagy a felhasználói dokumentáció alapján azonosíthatók.

12

Struktúra teszt

- A struktúrateszt esetében:
  - a teszteket a szoftver struktúrájának és implementációjának ismeretében készítjük.
- Ezt a megközelítést néha hívják ***fehér doboz*** tesztelésnek.
- Többnyire kis programegységekre, objektumokra, alprogramokra alkalmazzák.
- Az algoritmusról szerzett tudás alapján pontosabban tudjuk azonosítani az ekvivalencia-osztályokat.
  - Pl.: a keresőeljárás legyen a bináris keresés, ahol a sorozatot egy rendezett tömbként adjuk át.
  - A kód vizsgálatával láthatjuk, hogy a keresési teret három részre lehet osztani: középső elem, nála kisebbek és nála nagyobbak.

13

Tesztautomatizálás

- A tesztelés a szoftverfolyamat drága és fáradságos szakasza.
- Ennek eredményeképpen az elsők között kifejlesztett szoftvereszközök a tesztelő eszközök volfak.
- Ilyen keretrendszer a JUnit,
  - amely Java-osztályok olyan halmaza, amit a felhasználó kibővíthet annak érdekében, hogy létrehozzon egy automatizált tesztkörnyezetet.
  - Minden egyedi tesztet objektumként kell megvalósítani,
  - és a teszt végrehajtója futtatja az összes tesztet.
  - A teszteket úgy kell megírni, hogy jelezzék, hogy a tesztelt rendszer elvárt módon viselkedik-e.

14

KIALAKULÓ ÚJ TECHNOLÓGIÁK…

15

Szolgáltatásorientált architektúra

- A szolgáltatásorientált architektúra (service-oriented architecture, SOA) az elosztott rendszerek fejlesztésének módja,
  - ahol a rendszerek komponensei különálló szolgáltatások.
- Ezek a szolgáltatások földrajzilag egymástól távol elhelyezkedő számítógépeken futhatnak.
- A szolgáltatások kommunikációjának és az adatcserének a támogatására számos szabványos protokoll született.

16

Szolgáltatásorientált architektúra

- Következésképpen egy szolgáltatás lényege:
  - a szolgáltatás biztosítása független a platformtól,
  - az implementációs nyelvtől
  - és a szolgáltatást igénybe vevő alkalmazástól.
- **Szolgáltatás**: lazán csatolt, újrafelhasználható szoftverkomponens,
  - amely olyan diszkrét funkcionalitást zár be, amely elosztható és programozói eszközökkel elérhető.

17

Szolgáltatásorientált architektúra

- A www fejlődése:
  - azt eredményezte, hogy a kliensszámítógépek saját szervezetükön kívül elhelyezkedő távoli szerverekhez is hozzáférnek.
- Ha az információikat HTML-formátumra hozták, akkor azok elérhetővé váltak.
- Azonban ez a hozzáférés kizárólag csak webböngészőn keresztül volt lehetséges,
  - és az információtárolókhoz történő közvetlen hozzáférés nem volt megvalósítható.

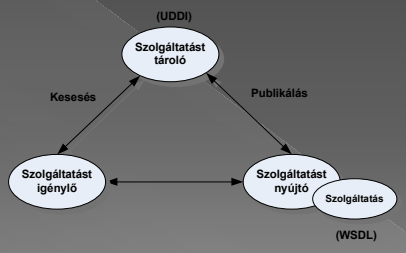
18

Szolgáltatásorientált architektúra

- Ennek a problémának a kiküszöbölése érdekében alakult ki a **webszolgáltatások** fogalma.
- A webszolgáltatás segítségével az információk elérhetővé válnak más programok számára is
  - egy web-szolgáltatás-interfész definiálásával és publikálásával.
  - Ez az interfész definiálja az elérhető adatokat, és megadja, hogy hogyan lehet azokhoz hozzáférni.
- **Webszolgáltatás:**
  - olyan szolgáltatás, amely szabványos internetes és XML-alapú szabványokkal érhető el.

19

A szolgáltatásorientált architektúra általános felépítése



20

Szolgáltatásorientált architektúra

- A szolgáltatások gyártói:
  - megtervezik és implementálják a szolgáltatásokat,
  - majd egy WSDL nevű nyelven specifikálják azokat.
  - Ezenkívül a szolgáltatásokról információkat publikálnak egy általánosan elérhető tárolóban az UDDI publikációs szabvány segítségével.
- A szolgáltatások igénybe vevői:
  - (néha szolgáltatáskliensnek is nevezik őket)
  - akik szeretnének használni egy szolgáltatást,
  - keresést hajtanak végre az UDDI-tárolóban, hogy megtalálják azt, valamint annak nyújtóját.

21

Szolgáltatásorientált architektúra

- Ezután alkalmazásukat összekapcsolhatják az adott szolgáltatással,
  - és kommunikálhatnak azzal.
  - A kommunikációhoz általában a SOAP nevű protokollt használják.
- A szolgáltatásorientált architektúra hatalmas fejlődést jelent,
  - különösen az üzleti alkalmazásrendszerek szempontjából.

22

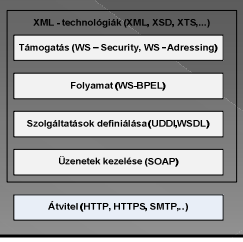
Szolgáltatásorientált architektúra

- **Oka:**
  - Rugalmasságot biztosít: a szolgáltatások elkészíthetők helyileg, illetve igénybe vehetők külső szolgáltatóktól.
  - A szolgáltatások bármilyen nyelven implementálhatók:
    - így lehetővé teszi, hogy a vállalat különböző részein esetlegesen használt különböző platformok és implementációs technológiák együttműködhessenek.
  - Ami a legfontosabb, a szolgáltatások segítségével a cégek és más szervezetek képesek az együttműködésre,
    - valamint egymás üzleti funkcióinak használatára.

23

Szolgáltatásorientált architektúra

- A szolgáltatásorientált architektúrák nem szenvednek inkompatibilitási problémáktól,
  - mert a fejlesztéseket a kezdetektől aktív szabványosítási folyamat követte.
  - Alapvető szabványok a webszolgáltatás támogatására:



24

Webszolgáltatás-orientált architektúrák szabványai

- **SOAP:** Egy üzenetcsere-szabvány, amely a szolgáltatások közötti kommunikációt támogatja.
  - Definiálja a szolgáltatások üzeneteinek szükséges és opcionális komponenseit.
- **WSDL:** Webszolgáltatás definíciós nyelv (Web Service Definition Language, WSDL)
  - definiálja, hogyan kell a szolgáltatások gyártóinak elkészíteniük szolgáltatásaik interfészeit.

25

Webszolgáltatás-orientált architektúrák szabványai

- **UDDI:** Az Univerzális leírás, felfedezés és integráció (Universal Description, Discovery and Integration, UDDI) szabványa.
  - Definiálja, hogy milyen komponenseket kell tartalmaznia a szolgáltatások specifikációinak,
    - hogy könnyen felfedezhetők legyenek.
  - Ez információkat tartalmaz a szolgáltatás gyártójáról, a szolgáltatásról,
    - a szolgáltatás leírásának helyéről.
  - Az UDDI-tárolók segítségével fedezhetik fel a felhasználók az elérhető szolgáltatásokat.

26

Webszolgáltatás-orientált architektúrák szabványai

- **WS-BPEL:**
  - Szabványos munkafolyamat nyelv,
  - különböző szolgáltatásokat tartalmazó folyamatprogramok definiálására használatos.

27

SZOLGÁLTATÁSOK TERVEZÉSE…

28

Szolgáltatások tervezése

- A **szolgáltatástervezés**:
  - az a folyamat, melynek során szolgáltatásorientált alkalmazásokban újrafelhasználható szolgáltatásokat fejlesztünk ki.
- A szolgáltatások tervezőinek biztosítaniuk kell:
  - hogy a szolgáltatás egy olyan újrafelhasználható absztrakciót reprezentál, amely különböző rendszerek számára is hasznos lehet.

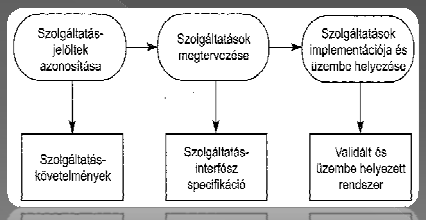
29

Szolgáltatások tervezése

- A szolgáltatástervezés folyamatának három logikai állomása van:
  - **Szolgáltatásjelöltek azonosítása:** Itt megkeressük azokat a lehetséges szolgáltatásokat, amelyeket a későbbiekben implementálhatunk,
    - és definiáljuk a szolgáltatások követelményeit.
  - **Szolgáltatások megtervezése:** Ebben a szakaszban tervezzük meg a logikai és a WSDL szolgáltatás-interfészeket.
  - **Szolgáltatások implementációja és üzembe helyezése:** Itt implementáljuk és teszteljük a szolgáltatásokat,
    - majd a felhasználók számára hozzáférhetővé tesszük őket.

30

Szolgáltatások tervezés folyamata



31

Szolgáltatásjelöltek azonosítása

- A szolgáltatásoknak az üzleti logikát kell támogatniuk.
- A szolgáltatás-jelöltek azonosítási folyamatának feladata:
  - megértse és elemezze a szervezet üzleti folyamatait,
  - majd eldöntse, milyen szolgáltatások szükségesek a folyamatok támogatásához.
- Alapvetően a szolgáltatásoknak három típusa különböztethető meg:
  - ***Segédszolgáltatások, Üzleti szolgáltatások, Koordinációs vagy folyamatszolgáltatások***

32

Szolgáltatásjelöltek azonosítása

- **1. Segédszolgáltatások:**
  - Ezek olyan általános funkcionalitást implementálnak, amit különböző üzleti folyamatok használhatnak.
    - Pl.: egy pénznemátváltó szolgáltatás, amelynek feladata egy pénznem (például dollár) átváltása egy másikra (például euró).
- **2. Üzleti szolgáltatások:**
  - Ezek a szolgáltatások egy bizonyos üzleti folyamathoz kapcsolódnak.
    - Pl.: üzleti szolgáltatás lehetne például a hallgatók regisztrálása egy kurzusra.

33

Szolgáltatásjelöltek azonosítása

- **3. Koordinációs vagy folyamatszolgáltatások:**
  - Ezek a szolgáltatások sokkal általánosabb üzleti folyamatot támogatnak,
    - amelyben több különböző faktor és tevékenység is megjelenhet.
  - Koordinációs szolgáltatásra jó példa egy vállalat rendelési szolgáltatása,
    - amelyben az ügyfelek leadhatják rendeléseiket és fizethetnek.

34

Szolgáltatásjelöltek azonosítása

- A szolgáltatásjelöltek azonosítása során cél:
  - olyan szolgáltatások kiválasztása, amelyek logikai egységet alkotnak,
  - függetlenek és újrafelhasználhatók.
- A folyamat végeredménye az azonosított szolgáltatások halmaza, valamint a hozzájuk kapcsolódó követelmények.

35

Köszönöm a figyelmet!

36