**RENDSZERMODELLEZÉS – MODELLEK ELLENŐRZÉSE 1**

**Követelmények modellekkel szemben**

Egy modell helyessége nem értelmes

->

Modellnek célja, kontextusa is van,

Modell szemben követelmények felállíthatóak.

**Követelmények funkcionalitás szerint**

Funkcionális követelmények

* Rendszer(összetevő) által ellátandó funkciót definiálnak
* Lásd: Developer funkcionalitása

Nemfunkcionális követelmények

* Rendszer minőségére vonatkoznak
  + Megbízhatóság
  + Teljesítményre vonatkozó kritériumok

**Biztonsági és élőségi követelmények**

Biztonsági követelmények

* Megengedett viselkedést definiálják
  + Engedélyezett és tiltott viselkedések
    - Univerzális követelmények, melyekre a rendszerre minden időpontban teljesülniük kell

Élőségi követelmények

* Elvárt viselkedést definiálják
  + Egzisztenciális követelmények, melyek szerint a rendszer megfelelő körülmények közt előbb-utóbb teljesíteni képes bizonyos elvárásokat

**Példák**

Egy jelzőlámpán egyszerre sosem világíthat a piros és a zöld fény. Ez egy biztonsági követelmény.

Élőségi követelmény, hogy a lámpa (előbb-utóbb) képes legyen zöldre váltani.

**Gyakori általános követelmények**

Holtpont vagy Deadlock

* Rendszer egy állapota, melyben a végrehajtás megáll, a rendszer többé nem képes állapotot váltani, és nem mutat semmilyen viselkedést
* Okok lehetnek:
  + Két, vagy több folyamat egymásra várakozik.
* Külső beavatkozás szükséges a holtpontból való kilépéshez
  + Következmény, hogy párhuzamos rendszereknél a holtpontmentesség gyakori követelmény.

Livelock

* Hasonlít a Deadlockhoz
* A livelockban résztvevő komponensek egy végtelen ciklusba ragadnak, amelyben nem végeznek hasznos tevékenységet.

Állapotgépek

* Determinizmus
* Teljesen specifikált működés

Folyamatmodellek

* Determinizmus
* Teljesen specifikáltság
* Jólformáltság

**Helyességvizsgálat**

Verifikáció

* Annak a vizsgálata, hogy az implementáció (az elkészített modell vagy rendszer) megfelel-e a specifikációnak
* Kérdések
  + Helyesen fejlesztjük-e a rendszert, megfelel-e az előírt kívánalmaknak?

Validáció

* Felhasználói elvárásokhoz hasonlítjuk a rendszert, azaz azt vizsgáljuk, hogy a megfelelő rendszert fejlesztjük-e.

Megjegyzések

* A megrendelőnek megfelelő és a specifikációnak megfelelő rendszer közt nem biztos, hogy azonosság van.

Felhasználói követelmények -> Specifikáció -> Implementáció

Implementáció –verifikáció- Specifikáció

Implementáció –validáció- Felhasználói követelmények

**Modellek ellenőrzése**

(

**Statikus ellenőrzés**

Statikus ellenőrzés

* A vizsgált rendszert vagy modellt annak végrehajtása, szimulációja nélkül elemezzük

Megjegyzések

* Hibák ránézésre látszódhatnak
* Hiba helye könnyen beazonosítható
* Dinamikus módszerek esetén hiba felismerése és okának megtalálása két külön feladat.

Alkalmazások

* Szintaktikai hibák keresése
  + Ekkor a rendszer tipikusan nem is futtatható

1. Szintaktikai hibák vizsgálata
2. Szemantikai hibák vizsgálata

**Szintaktikai hibák vizsgálata**

* Azon hibák, amelyek következtében egy modell nem felel meg a metamodelljének, vagy egy program nem felel meg a használt programozási nyelv formai megkötéseinek.
* Példák
  + Állapotokhoz nem kötött állapotátmenet egy állapotgépben
  + Hiányzó zárójel egy programban
* General
  + Szintaktikai hibák nagy biztonsággal kimutathatók
  + Ritka, hogy egy statikus ellenőrző helyes kódot, vagy modellt szintaktikailag hibásnak értékelne.

**Szemantikai hibák vizsgálata**

* Szintaktikailag helyes rendszer, viszont nem értelmes vagy nem elvárt módon történő viselkedés
* Code Smellnek nevezzük a gyanús kódrészleteket
  + Gyanús egy kódrészlet, ha a fejlesztő szándéka eltér az implementációjától

**Védekezés szemantikai hibák ellen**

Null

Hibaminták

* Statikus ellenőrző eszköz keresi meg őket
* Ha egy decision elem egy join elemmel áll párban, akkor az rossz
  + Ha egy modell rossz, akkor javítsuk.

Erősebb megkötések alkalmazása

* Szintaxis és formalizmus
  + Egy nyelv szintaxisa elemeire vonatkozó formalizmussal szemantikai hibák elkerülhetőek
* Kódolási szabályok
* Jólstruktúrált folyamatmodellek
  + A modell kizárólag adott mintákból állhat elő
    - Példa egy halmaz mintára
      * {üres folyamat, elemi tevékenység, szekvencia, ciklus, döntés, párhuzamosság}
    - Ezzel a szintaxist úgy kötjük meg, hogy a tipikus szemantikai hibák ne állhassanak elő

Szemantikai hiba jelentése

* Függ a modellezési, vagy programozási nyelvtől
* Függhet alkalmazási területtől, vagy konkrét felhasználástól
  + Egyes alkalmazási területeken további tervezési szabályok definiálhatóak, melyek tovább korlátozzák a modellezés vagy programozás szabadságát.
    - Példa: Biztonságkritikus programok esetén gyakran tiltott a dinamikus memóriafoglalás.
    - Ilyen esetekben kiegészíthetjük a hibaminták készletét a malloc és hasonló konstrukciókkal.

**Szimbolikus végrehatjás**

…

Milyen invariánsai lehetnek egy rendszernek?

Minden rendszer invariánsai különbözőek?

Mik határozzák meg egy rendszer invariánsait?

-o-

**Tesztelés**

Tesztelés

* Olyan tevékenység, amely során a rendszer ( vagy egy futtatható modelljét) bizonyos meghatározott körülmények közt futtatjuk, majd az eredményeket összehasonlítjuk az elvárásainkkal.
* Cél: A vizsgált rendszer minőségének felmérése és/vagy javítása azáltal, hogy hibákat azonosítunk.

Alapfogalmak

* Tesztelendő rendszer (SUT)
  + Az a rendszer, amit teszt során futtatni fogunk a vizsgálat céljából
    - Pl.: M
* Tesztbemenetek
  + A tesztelendő rendszer számára biztosítandó bemeneti adatok
    - Pl.: Használati eset táblázat
* Tesztorákulum:
  + Olyan algoritmus és/vagy adat, amely alapján a végrehajtott tesztről eldönthető annak eredménye
    - Pl.: Végeredmény identitását meghatározó objektum
* Tesztesetnek
  + Azon adatok összessége, amelyek egy teszt futtatásához és annak értékeléséhez szükségesek.
  + Bemeneti értékek, végrehajtási előfeltételek, elvárt eredmények (elfogadási kritérium) és végrehajtási utófeltételek halmaza, amelyeket egy konkrét célért vagy a tesztért fejlesztettek
    - Tesztesetnek egy helyessége elvégezhető munkához kell hogy vezessen
    - Mérnie kell egy rendszer minden komponensét, mely kapcsolatban van egy használati eset elvégzésével.
* Tesztkészlet
  + Tesztesetek egy adott halmaza
* Tesztfuttatás
  + Egy vagy több teszteset végrehajtása
    - Az orákulum segítségével tesztfuttatás során megtudjuk a teszt eredményét
      * Az eredmény lehet sikeres, sikertelen, hibás
        + Utóbbi esetben nem eldönthető a tesztről, hogy sikeres-e vagy sem
        + Példa.: Tesztrendszerben hiba történt és a SUT helyességéről nem tudunk nyilatkozni.
      * A kapott és a tesztesetben megfogalmazott elvárt kimenetek összehasonlításával kapható meg a teszt eredménye
      * A teszt eredmény származhat egy referenciaimplementációval összehasonlításból, vagy ellenőrizhetünk implicit elvárásokat, például azt, hogy a kód nem dob kivételt

**A tesztek kategorizálása**

Modulteszt (Komponensteszt, egységteszt)

* Az a teszt, amely csak egyes izolált komponenseket tesztelnek

Integrációs teszt

* Az a teszt, amelynek célja az integrált egységek közötti interfészekben, illetve kölcsönhatásokban lévő hibák megtalálása

Rendszerteszt

* Az a teszt, amelyben a teljes, integrált rendszert vizsgáljuk annak érdekében, hogy ellenőrizzük a követelményeknek való megfelelőséget.

Regressziós teszt

* Módosított rendszer utáni szelektív ujratesztelés
* Ortogonális kategória, mely egységeket vagy teljes rendszert is vizsgálahat.

**A tesztelés metrikái**

Általános cél

* A vizsgált rendszer minőségének javítása hibák megtalálásán és javításán keresztül.
* Egy idő után a tesztelés folytatása nem célszerű, mert a rendszer minősége már "elég jó".
* Honnan tudjuk, hogy mikortól lesz egy rendszer "elég jó"?

Tesztkészlet fedésének mérése

* Tesztfedettség alapötlete
  + Ha a tesztkészlet egyik tesztesete sem látogat meg egy adott állapotot egy állapotgépben, akkor az az állapot biztosan nem lesz vizsgálva
    - Emiatt a nem vizsgált állapot minőségéről nem vonhatunk le következtetést.
  + A fentihez hasonló állítás elmondható egy metódus hívásával kapcsolatban is
  + Megjegyzések
    - Ha egy tesztkészlet meglátogat minden állapotot, vagy meghív minden metódust, nem mondhatjuk el, hogy mindent megvizsgáltunk.
      * Minden állapot érintéséből nem következik minden állapotátmenet érintése
      * Minden metódus meghívása nem feltétlenül követi minden utasítás meghívását.
* Fedettségi metrikák
  + Állapotfedettség
    - Egy adott tesztkészlet által (bejárt) állapotok és az összes állapotok arányát adja meg.
  + Átmenetfedettség
    - Egy adott tesztkészlet által bejárt állapotátmenetek és az összes átmenetek arányát adja meg
  + Egy vezérlési folyamban utasításfedettség egy adott tesztkészlet által érintett utasítások és az összes utasítások arányát adja meg.

**Tesztelés futásidőben**

Elvárások elhelyezhetőek egy rendszerben és folyamatosan vizsgálhatóak.

Minden állapotban elvárt követelményeket invariánsoknak nevezzük

A monitorozás lépései

* Bemenetek ellenőrzése
  + Bemeneti adatok megfelelősségének vizsgálata a definiált bemeneti invariánsok alapján
* Hihetőségvizsgálat
  + Kimeneti adatok megfelelősségét vizsgáljuk a bemeneti adatok és a definiált kimeneti invariánsok alapján.

x implementációjának részei

* Előfeltétel vizsgálata
  + Ha ez nem teljesül, akkor kivételről beszélünk
  + Ha az előfeltétel nem teljesül, annak oka a rendszer hibás használata (nem megfelelő bemeneti adatokat kapott)
* Amennyiban az előfeltétel teljesült, megtörténik az érdemi logika végrehajtása
* Végrehajtás után az utófeltétel vizsgálatára kerül sor.
* Amennyiben az utófeltétel nem teljesül, olyan hibás állapotba került a rendszer, amely kezelésére nincs felkészítve. Ennek oka lehet a hibás implementáció vagy futásidejű hiba.

Konklúziók:

Lassan neki kell hogy álljak fejleszteni azokat az alrendszereket, melyek a csapat tagjainak a funkcionalitását írják le.

* Pl a rendszer S kap egy u bemenetet, ami egyértelműen a Scrum Master-hez kerül, azaz a Scrum Master-hez előbb-utóbb eljut egy u bemenet.
  + Az u bemenetre vonatkozó kérdések:
    - Milyen típusa van u-nak
    - Scrum Master milyen kimenetet ad?
    - Ha az u bemenet T típusú, akkor egyértelmű annak a feldolgozása?
      * Egyértelműnek kellene lennie
      * Hogyan kapcsolódik egy bemenet feldolgozásának milyenségéhez a determinizmus fogalma?
* Ehhez szükséges követelményeknek egy halmaza, melyeknek meg kell felelnie a fejlesztendő rendszernek, vagy a szkriptnek
  + Milyen paraméterek jellemzik az ETL folyamatot?
  + Az ETL folyamatnak milyen lépései vannak?
  + Az ETL folyamat lépései során milyen magasszintű problémákkal kell szembenézni?
  + Milyen rendszerek működése szükséges a magasszintű problémák megoldásához?

Rendszerek megengedett és tiltott viselkedése adott t időpillanatokban, vagy T intervallumokban.

**System**

(

1. Description
2. Analysis
3. Test if Analysis is good, otherwise go to Description
4. Implementation if Test is good, otherwise go to Description

)