Szoftvertechnológia

jegyzet

BME Szoftvertechnológia tárgy 2008/2009 1. félévében az előadásokon vetített diákhoz készült kézzel írt kiegészítések.

Tartalomjegyzék

Szoftvertechnológia

2008.09.10.

- László Zoltán IB 323

ptuser @ fsarkas.iit.bme.hu

fsarkas.iit.bme.hu/~ptuser/ - diák itt + feladatok, ZH,-k

Vizsgák: jan 6, jan 13, jan 27

HF nov 25-ig kell kódot

URAL-t rendbeszedni mp 767

- follal kell kitölteni a HF-t

(2) Part I (31-50) 2008.09.12

Thurs -25

(3) Part II (1-18) 2008.09.17

- phased - life-cycle } életciklus modell
  other
- manufacturing - gyártás

- Minőség
  - a felhasználó mindig elégedetlen, ami jobb
  - a gyártás minőség - nem a tesztelésen múlik (Bácsi hiába tesztel nem lesz BMW!
  - termékek gyártanak, amelyik jobban bírja az a jobb
  - ár-érték viszony - az ár jelent valamit

- szoftverrel a gyártás a befolyásoló minőségi tényező
- minőségi szoftver minőségi folyamattal állítható elő

B/Poly

Absztrakció: – elvonatkoztatás

- a részletek elrejtését jelenti
  - nagyvonalúvala dolgozz a kézzelfogható magnó
  - apróbb felvenni is tudsz vele
  - a gyerek már programozni, beállítani is tudja
  - a szerelésnek bonyolultabb elképzelései vannak
  - a gyártó többet tud
  - a legtöbbet a tervező tudja

Formalizáltság:

- előírások mennyire szigorúak, ezeket mennyire tartjuk be
  - a szavak a beszédben nem szabályosak
  - mondatokat erre szabályosabb
  - ürlap már formalizáltabb – hamarabb feldolgozható → gépi transzformáció
  - a matematika – végtelen formalizmus

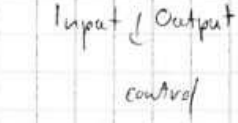
7. dia – szaggatott a jó út

- először specifikáljuk ------>
- majd tervezünk ↓

8. dia

implementációs függőség – a vízszintes tengely felirata

9. dia



12 db a

Folyamat-leírás
- cél
- szereplők / felelősségük
- mikor lehet ellenőrizni a folyamatot? – folyamatkezdési kritériumok
- bemenet
- Folyamat tevékenységeinek leírása
- kimenet / eredmények
- mikor hogyan ábrázolja?
- Folyamat – mérés (metrics)

13. Szoftver process
- emberi – munkaigényes
- csak megy és uralkodj!

14.
- a szoftverekbe ~~nagy része~~ még megvalósítás előtt jellemek
- vissnat csinálunk valamit, a terv vissza

Vízesés modell
- minden lépés után vizsgálat, van és vissza lehet/kell lépni

18. ellenőrizendő dokumentumok listája

(4.) Part (18- 2008.09.12

18 • Követelmények
- Specifikáció
- tervezés – a specifikált szerkezeti építőelemeinek tervezése pl: építőanyagbeszerzés
- megvalósítás

1, rendszerdefiníció
- állnak leírása

projekt terv
- hogyan lehet a projektet végrehajtani

FFR – Project Feasibility Review – megvalósíthatósági értekezlet

2, • formális leírás arról a rendszerről, ami kielégíti a követelményeket
- előzetes felhasználói kézikönyv
- előzetes ellenőrzési terv – hogyan, mikor kell bizonyos ellenőrzéseket végezni
SRR – meeting

3, • architektúra – mik a fő elemek és hogyan kapcsolódnak egymáshoz
- miből is áll a szoftver – függvények, dokumentumok?
- perzisztencia – adatok megőrzése, hogy ki/be tűl, amikor kikapcsol a gép
- elosztott működés megvalósítása
- felhasználói felület – milyen ablakok-rendszert használunk
- Preliminary Design Review meeting

4, Részletes terv
- melyik programnak mit csinál
- végleges felhasználói kézikönyv
- Critical Design Review – meeting

5, megvalósítás
- elkezdenek programozni
- code-review – hibá-átellenőrzés
- átadási teszt / üzembehelyezési terv – be kell tanítani a stábot
- virrga a stábnak

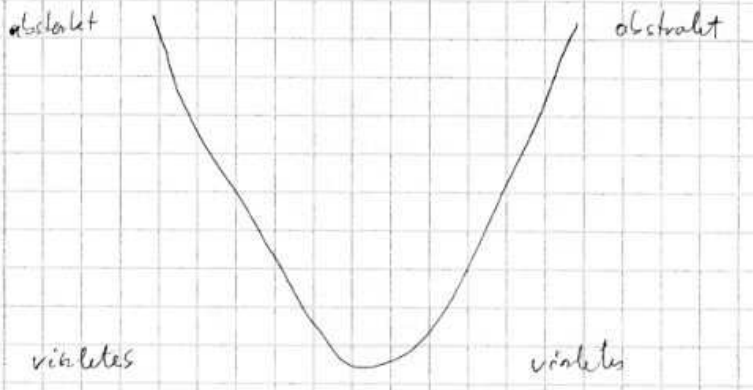
SCR – Source Code R
ACR – acceptance Code R
} meeting

6, Ez az a szoftver amit kellett?.
- mindent átnézni, javítani, ami változott

PRR – átadás előtti meeting
PPM – átadás utáni –"–

19 V-modell

- Tesztelissel kapcsolatos a jobb oldal
  - össze kell vakni az egyes metódusokat software – integráció



- egységek tesztje – maga tesztelj, alsó úton
- ellenőrzik összevakni az egységeket – pl autóból a fékrendszer
- rendszer tesztje – maga az autó

Validálás:
– összességében ilyen autót akartam-e / kívánni vele az utat

20. ~~Magyarázó kor~~

21. iterációs modell

- prototípust csinálni, majd javítgatni
  - első körrel az elvárásnak és csak a 3. legyen a sajátod

22: RAD

- iterációs modell egyik fajtája

23. Spirál modell

go – no go pont / dönteni kell, hogy megvalósítható-e

24. képesség érettségének modelje

- szoftverek csillagozása - mint a hotelekénél — Level CMM 3 company

pl: Evosoft ilyen

Carnegie Mellon University - hasonló helyes a szoftvereknél

26

1, kezdeti szint

2, ismétlődő szint

- van definiált folyamat

3, definiált

- van előírás és ez standard (szabványos)

4, ∞ mért folyamatok - mérik a folyamatokat

5, van-e a folyamat standard / hanem az előírt folyamat is hangolható

- módszeres optimalizálás

27: - kövvonalaszatlan folyamat

28: - van definiált folyamat - vannak bizonyos meghatározott pontok

29: - a nagy dokumentumok szabványos belső vázzal vannak

30: - számszerű mérések, hogy mi készült el

31 32 real-time - szigorú időbeli követelmények (nem feltétlen nagyon gyors)

Part III (1—

A menedzsment

- a fejlesztés nagy része együttműködés
- overhead - kétértelműség, pluszköltség

pl főzés során gyalulás, vagy vezetni a világítás

- bevesd el jódo embert
- ha valaki nem fér be port ki kell dobni
- „ha egy projekt kiesésben van minden további kezdődő ember kiesést jelent" – Brooks – törvény
- ! a nélkülözhetetlen embert ki kell rúgni !
  - lépjen kiszállítsa a projektet
  - le kell vonni vele

Szoft tech

(5)

2008.09.24.

6. óra Irányítás, softwareprojekt irányítása

- milyen erőforrások, költségek, ütemezés kell
- a projekt tervet kell úgy időközönként felül kell vizsgálni/pontosítani

- Project manager a felelős a projektért
  - állandóan újra kell allokálni az erőforrásokat, ütemtervet módosítani, követelményeket átvenni, új dolgozókat tanítani a stábot

(8) • hosszú távú terv / rövid távú terv

(9) Erőforrások:

(1) ember – egy főre 2-3 millió bevétel kell jusson

(2) hardver – ma már eltörpül

(3) software emberöröl

- pontosítja softverek
- másgép programját nem lehet másgépen szimulálni – bonyolult rendszereknél kell szimulátor, emulátor

(10) Mit kell menedzselni?

- időt: ha nincs munkája az alkalmazottaknak → tönkremegy a cég
- információt: dolgozókat el kell látni információkkal
- szervezet:
  - mágikus hetes – általában 7 sőt lehet megjegyezni, ha ennél több dologgal szabályok
  - Több ember nem tud jól együtt dolgozni

11. dia Ütemezés

- lehet kapni komplett projekt-template-et
- projekt felvétele
  - egymásra épülnek, van amit lehet párhuzamosan, vagy vannak függőségi elvárások

(14)

(15) elemi feladat, amivel előrehaladhatok a projektet

Task

~~1-3-9-xxxxx~~ 1-3-9-11-12 - kritikus út

Ez egy Gant-ábra!

(16) Vocházatok

(17)
- stabil oktatás
- hardver nem kerül elérhető
- túl sok változás a követelményekben
- műveket, valósítást alulbecsülték
- technológia túlhaladott vagytanúk
- ~~xxxxxx~~ :

(18) Kockázat-menedzsment! ← (ZH)-ra

1, kockázat azonosítása

2, elemzés

3, tervezés

4, monitorozás

(1) Kockázatazonosítás

- kockázat felvonulása
  - technológia
  - emberekkel kapcsolatos
  - szervezeti
  - követelményekben rejlő
  - becsléssel kapcsolatos

} kockázatok

(2) Kockázatelemzés

- kockázat – valószínűség – hatása

(3) Kockázatkezelés (ZH)

- 3 stratégiai lépés

a, hogyan tudjuk elkerülni a kockázatokat – pl.: jól fizetni a csapatot, hogy ne menjen el

b, kockázat hatásának minimalizálása

c, ha bekövetkezett, akkor hogyan tudunk felállni

(4) Kockázatfigyelés

- folyamatosan figyelni kell a kockázatokat

(23) Becslés

- a pontosság <u>inkább igény</u> a projekt minőségével nő
- a valóságban ez pont fordítva megy

(24) - ellenőrizhetőnek kell lennie a becslésnek

(25) - megismételhetőnek kell lennie

- ha már volt ilyen típusú projekt, akkor abból indulunk ki
- ha új van, amit még nem csináltunk
  - egyéni standardok
  - számítások

(26) Mit becsülünk

- Mekkora a feladat/ mennyire bonyolult
- szoftver mérete – Line of code (LOC), Function point analysys (FPA)
  - funkció-pont – <u>alkalmat 2</u>

(27) - akkor az évtizedek vannak – kb (10)-es <u>1 funkciópontú</u> program van.

(100 000) Ft

kb 100 sor program

(28) Program & produktivitás

Projekt terv - nem kell feltölt

1.0
:
:
7.0

1.1 Project scope
- ki van kívül, ki van belül

1.2. Tűz stb

Követelménygyűjtés (4)

- a hibák nagyobbik felét tesztelésnél fedezzük fel
  - az idő kb 56 %-t a követelménygyűjtésnél valósítsuk el

(5)
- inkorrekt tényezők 50%
- elhanyagolás 30%
- ellentmondás 15%
- többértelműség 7%

"Megetette a lovat a zabbal" -:)

2008.09.26.

(6)

Part IV (3 - .)

- Követelmény definíció
  - kvázi hosszú hangvételű - emberi nyelven
- specifikáció - formalizált

Funkcionális követelmények – mit kell tudnia

Nem – " – – a Hibátlanság

(7) External – külső környezettel kapcsolatos

Használhatóság: – telepítésnél

– memória hely

Hordozhatóság:

Szerencsét: –

(18) (19)

Történetiség az adatok közt! – History

- korábbi időszaki jelentésekben gyűjtött adatot

(14) Hogyan gyűjtjük az információkat?

- interjúk
- kérdőíves rendszer
- Zsiga kérdések megírják a listát – az automatizálásnál is úgy kell, hogy kelleti legyen ∈ ha nincs ilyen kérdőség, akkor elkészítjük majd az Zsiga kérdéseket a gépet
- nem csak a főnökökkel, hanem a beosztottakkal is kell tárgyalni

(16) 1.0 Bevezetés – rendszer körvonala, mi-kell-van

2.0 funkcionális és adatkövetelmények
- interfészek

3.0 alrendszerek leírása

4.0 modellezési, tőlalakításos paradigmák

5.0 Kapcsolat a project-el

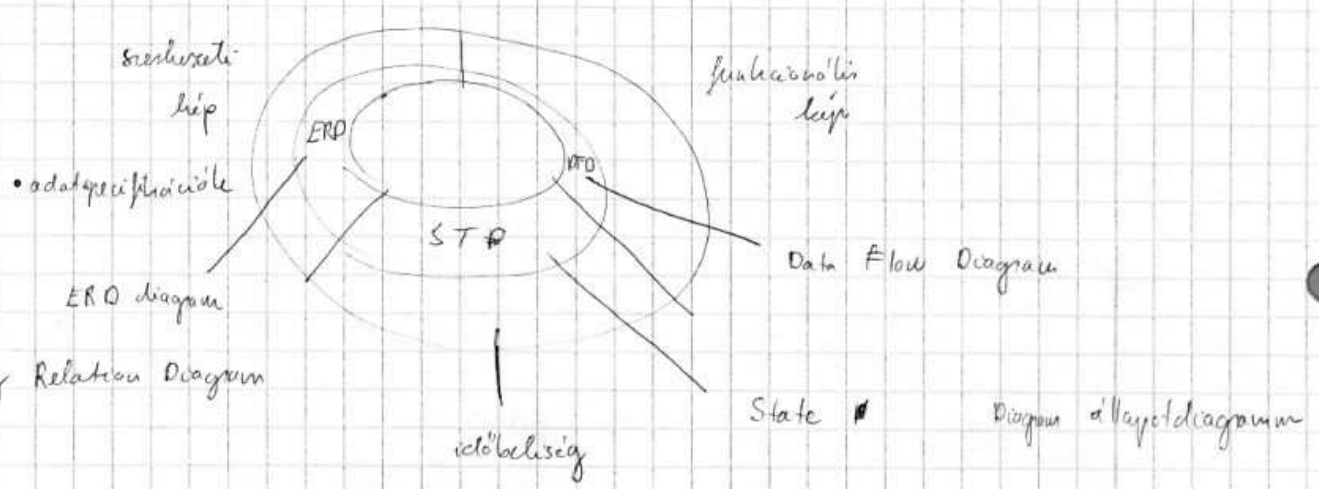
6.0 Függelék
- célszerű felsorolni a felhasznált anyagokat, irodalomjegyzék
- szómagyarázatok

Specifikáció Part V

- célja olyan minőségi formális leírása, ami kielégíti a követelményeket

(4.)

- szerkezeti lista
- tevékenységi lista
- dinamikai lista – a végrehajtás sorrendje



(5) funkcionális modell

DFD – adatfolyam ábra

- adat folyam →
- folyamat ○
- state tároló ═══
- adat forrás, vagy nyelő □ TERMINATOR

(7) Környezeti (context) diagramm – az egész folyamat egy gombóc, nincs adattár

Mozijegy – eladás: change – visszajáró

(8) az alsó része a felső gombóc tartalma

- vagyon annyi tulajdonos és tulajdonság kell - szintek közötti kiegyensúlyozottság

(9) vonalkód eladás helyén

(10) Adattároló

- csak folyamatokból nyerhetünk
- context diagramon nem lehet
- több szinten is lábadhat
- a folyó idő is adattárban van
- írás
  - annyiban jogosult olvasni is folyamat ezen az íráshoz kell
- olvasás
- írás-olvasás

- szekvencia
  szelekció
  iteráció
  } - ez vagy az if/switch
  - ismétlés

(7)

2008.10.01.

Specifikáció: formális leírás egy nem létező szoftverről

blueprint - műszaki, szakmai rajz

(12) flowchart - folyamatábra

(13) NS-diagram

- nem lehet oda-vissza ugrálgatni

Feladat

A mozi előadásait az előadás jellemzőinek (film címe, előadás időpontja, helyárak) megadásával a mozi vezetője definiálja. A pénztáros a definiált és még be nem fejezett előadásokra bérpénz fizetése ellenében jegyet ad / árusít, amely egyben számla is. A jegyen rajta van a mozinak az adózással kapcsolatos adatain kívül a film címe, előadás időpontja, az elfoglalt székek azonosítója (erkély, bal, 2-sor 2-5 szék) és a jegy ára.

Egy jeggyel több széket is foglalhatunk. A pénztárosnál telefonon előfizetők előjegyezhetnek jegyet.

gombóc = tevékenység — ige, igenév

Főnév

igék

- ~~vagyok~~
- definiál ???
- ~~definiált~~
- ~~befejezett~~
- ~~befejezett~~
- fizet ???
- ad (jegyet) ✓
- foglal
- előjegyez

terminátor végzi

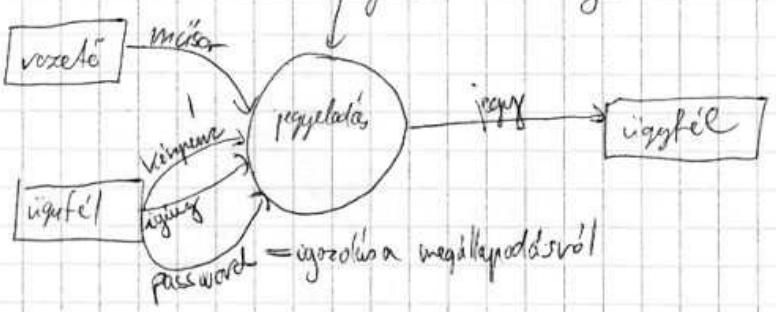
főnevek

- ~~mozi~~, előadás, ~~jellemző~~
- ~~film címe~~
- ~~előadás időpontja~~
- ~~helyárak~~
- ~~pénztáros~~
- vezető ✓ — ő a műsor forrása
- ~~pénztáros~~ — ő csak motor, semmiről nem dönt — gép is lehetne helyette
- bérpénz ✓
- jegy ✓
- széle ✓ — arról megy a harc
- ~~telefon~~
- ~~jegyfél~~ — az is kell

- ~~helyárak~~ árai adatállomány — egyes sorok mit jelentenek abban a környezetben

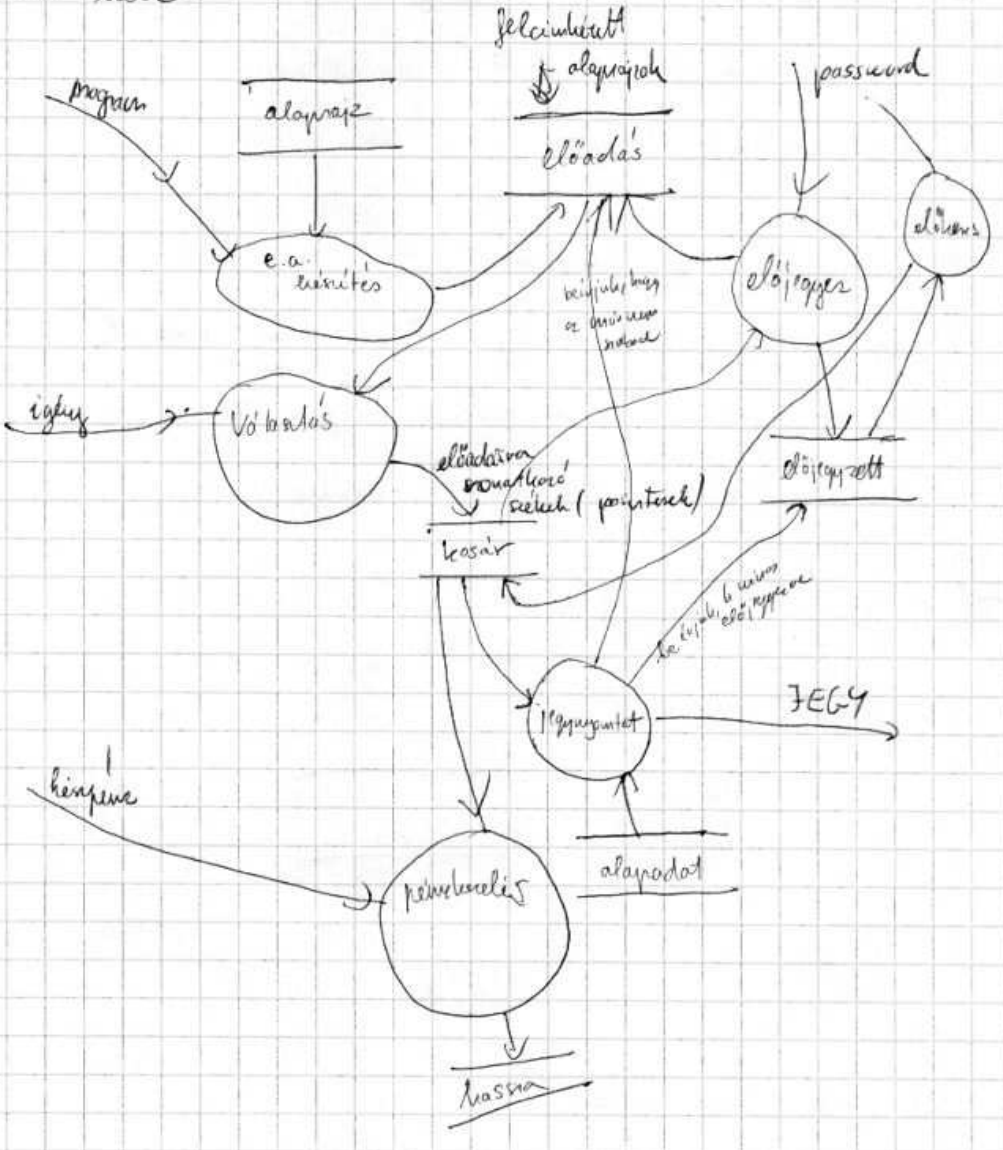
Context diagram

gombóc csak ige



(• amit a TERMINÁTOR vége, ott valamiből a túloldalon valamit csinál)

o kezdeni kell



• olyan miket egy áramkör NEM BÍR

  • időben nem dönthető el, hogy két-miket kívánnak

  • majd kell állapotgép, idődiagram

(8)

2008. 10. 08.

Specifikáció - 3-féle

① funkcionális kép

② szerkezeti kép – szereplők, kapcsolatok

③ viselkedési kép

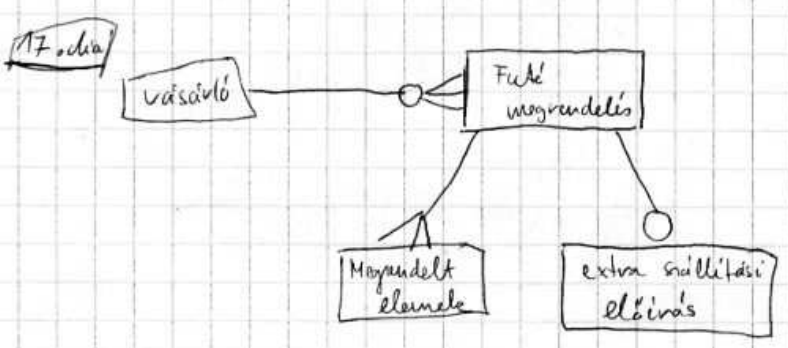
Szerkezeti lép

Főszereplők: - az adatok, amik a világ valamilyen dolgát leírja → (ember, személy)

- elemi attribútumokkal felruházott
- kapcsolatok az adatok között

ERD – Entity-relationship Diagram

- cardinality – egy példánynak, hogy másik entitás tartozik
- modalitás – kötelező-e a relációnak jen állnia
  - ha van 0, akkor opcionális



(19)

- szekvencia
  - adat = attribútumok összessége
- reláció – unió
- iterátor – tömbök, listák

– komplex elemek – a jövő 3-ból

– kulcs megjelölése: @, #

(20) XML

(21) • nincs átlapolás – korrekt skatulyázás

(23)

(25) • másodlik a jó

• attribútumnak pl ~~az~~ sorszám lehet – nem a személy tulajdonsága

↑ pl ami a Te nyilvántartásodban kell

- XML Tutorial W3C school

28

Parse -olt

Parse Code ? ? ?

\+ – egyszer, többször

* – 0-szor, ~~egyszer~~ többször

REQUIRED – kötelező

parsolni = feldolgozni

Viselkedési lap

- a külvilág eseményeire reagálni kell
- itt is DFD diagram van

- - - -> vezérlő jelek szaggatottal

⇓

a kommunikációs sík

35

állapot	bekötött	szeparátor jött	újsor jött
⋮			

ÁLLAPOT-TÁBLA

u: ciklok , növelem a sort

x = c=0

t1 buffer[c++] = c1

t2: if (i>10) | print (u, buffer)

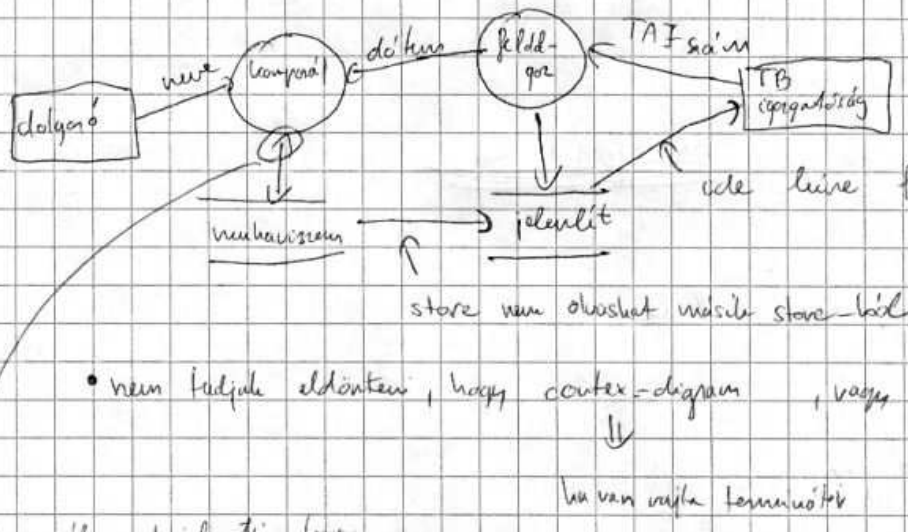
sor sió

az igazából ka.

2008. 10. 10. (9)

(1)

Miért vesse az ábra?



• nem tudjuk eldönteni, hogy control-diagram, vagy data-flow ábra

⇓

ha van vajla terminátor

⇓

store tárolt gombóc

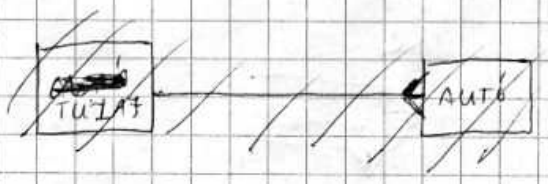
az arról ad jelentést, hogy a

folyamat azért olvas a munkaviszonyból, mert ez egy kimenethez kell

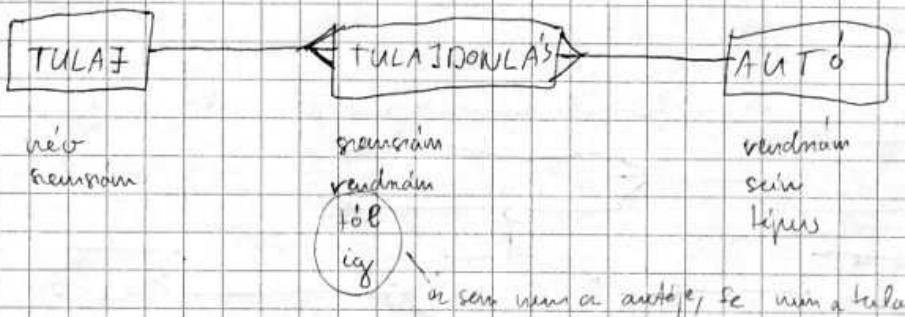
→ most nincs kimenete!

(2) Olyan nyilvántartást alkalmas készíteni, amellyel kivethető, hogy egy adott időpillanatban ki (volt) az autó tulajdonosa. Egy autónak egyidejűleg csak egy tulajdonosa lehet!

Rajzolni kell egy entitáskapcsolati diagramot

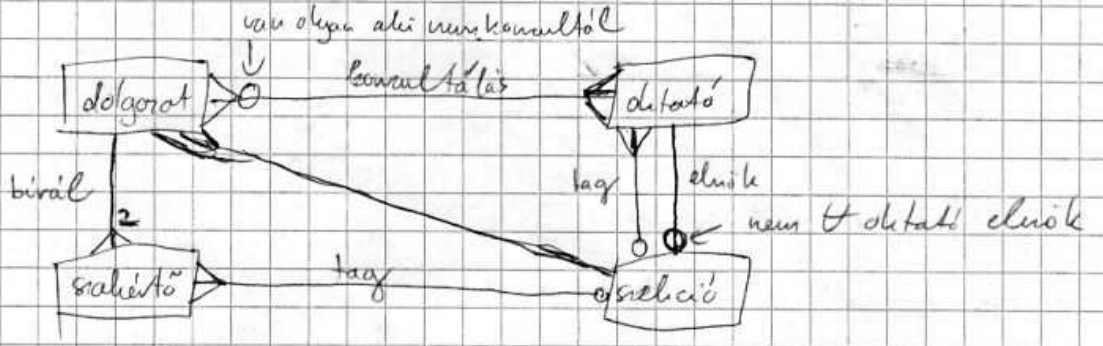


- autónak mindig van tulajdonosa
- a régi autókról is tudni kell

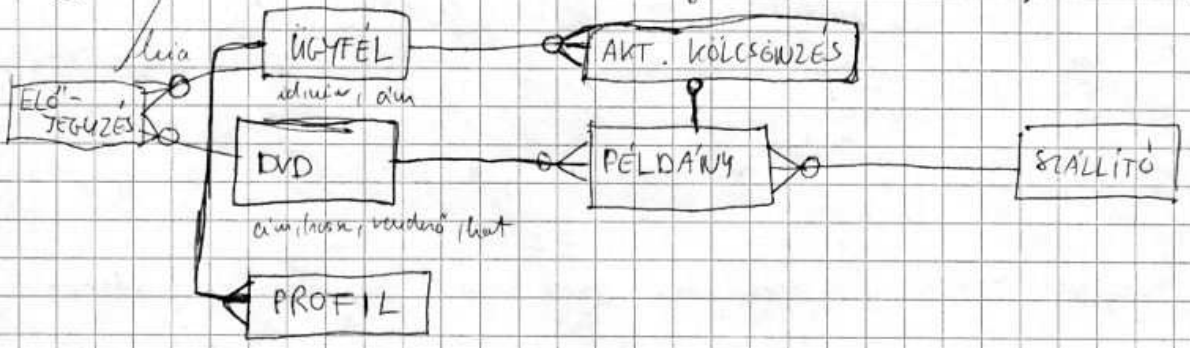


- orak bővülhet → ha tűvennünk belőle valahol lenne olyan, hogy nem tudnánk megmondani egy bizonyos időpontban ki volt a tulaj

(3) A Nari TDK konferenciára benyújtott dolgozatok oktatók bevonásával bírálhatók. Egy oktató több dolgozatot is konzultálhatott, és van olyan dolgozat is, aminek több konzulense is van.
Ü dolgozatot pontosan 2 egyetemen kívüli szakértővel bíráltatnak el. Egy szakértő csak egy dolgozatot bírál. Minden dolgozatot egy névvel ellátott szekcióba sorolnak, egy szekcióban 7–10-t. A szekció munkáját az elnök irányítja, aki egyetemi oktató. A szekciók munkájában szekcióelnökként vett vesz még legalább 1 oktató és legalább 1 a bírálásban is részt vevő szakértő!



(4) 1/ videó feladásakor a DVD-ről nyilvántartják a címét, a kiadóját, a kiadás



id, hat, db

(5)
- nincs gyökere

(6)
- enter az <xxx/> után ← de egyébként jól formált

(7)
- változónál "-jelbe

(8)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE x [
    <!ELEMENT x (d|b)>
    <!ELEMENT d (a, c*)>
    <!ELEMENT a (#PCDATA)>
    <!ELEMENT b (#PCDATA)>
    <!ELEMENT c (#PCDATA)>
]>
```

⇒

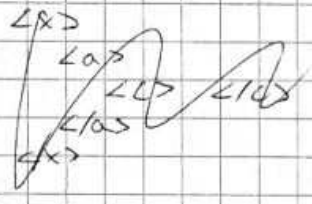
```
<x>
  <d>
    <a>.. aa </a>
    <c> .. c </c> ← akárhol többször
  </d>
</x>
```

(9)

```
<x>
```

x
d (b*, (c|d))
d
b ① vagy
c

```
</x>
```



(10) $x = a + \{b + [a \mid c]\}$

$\langle !\ ELEMENT\ x\ (a, (b, (a|c))*) \rangle$

(11)

- középső nem jó szintaktikailag
- a 3. a jobb

(12)

- állapotátlátót szimulálni oszlopok száma: 3 oszlop (tudor, vidor, kutka)

  Sorok száma: ~~nem~~ nem 2 sorral kell lenni

  proc, get, write

  data - store nem kell!

(10)

2008. 10. 15

Part V (37 -

Specifikáció:

- formálisan leírni egy szoftvert ami kielégíti a követelményeket

(37) Szintaxis-gráf

- a körben konstansok vannak
- ami nem kerek dobozban van, az változtatható, helyettesíteni tudjuk
- lebegőpontos számot definiál az alsó ábra
  - egy számról ellenőrizni tudjuk, hogy helyes-e

(38) BNF – algebrai leírás

::= helyettesítés jele

digit ::= 1|2 ← konstans

int ::= <digit> ← nem terminális szimbólum, helyettesíthető

(39) Algebrai axiómák

- absztrakt adatokat axiómákkal absztrakt leírni
- definiálni kell:
  - halmazokat
  - művelet prototípusokat – szignatúra
  - axiómá – kifejezés = kifejezés

(40) Stack:

- halmazok: stack, item, boolean
  ↑ amit bele lehet tenni
- műveletek: Neki – új stacket ad
  változtatnak → push(stack, item) – beletesz
  pop(stack) – legutoló elemet leszedi (kiveszi)
  visszatérés típusú művelet → top(stack) – megmondja mi van a tetején ← read-only művelet
  empty(stack) – igaz, ha üres

(41) – axiómák:

EMPTY(NEW()) = true; ← a frissen született stack üres

EMPTY(PUSH(s,i)) = false;

TOP(NEW()) = undefined;

TOP(PUSH(s,i)) = i

POP(NEW()) = NEW(); ← üres stackből csak üresebb

POP(PUSH(s,i)) = s; ← a többi elem változatlan!!! marad

(42)

push(pop(push(push(new(1,3), 12), -1)

s — i

a zárójelek átalakíthatók!!

push(push(new(1,3), -1)

a) Konstruktor-műveletek:

- azok a műveletek, amivel a világ ∀ elemje előállítható
- push, new

b) Módosító-műveletek (modifier)

- pop
- konstrukcióban szokták alkalmazni

(43) Array

- array, int, item

- New (int, int) → array
- Assign (array, int, item) → array
- Bound (array) → int, int
- Eval (Array, int) → item

Axiómák:

Bound (new (x,y)) ⇒ x,y

Assign (a, u, v) = Bound (s)

Eval (New (x,y), u) = undefined — új tömbben nem tudom mi van

Eval (Assign (a, u, v), w) =

if (u=w) v else Eval (a, w)

- ha beraknak egy új elemet, akkor a többi nem változik

(45) **Lista** (FIFO)

- list, int, int
- CRT() → list
- ADD(list, item) → list
- TAIL(list) → list ⇒ a legrégebbi elemet eldobja, a többi marad
- HEAD(list) →
- LGTH →

HEAD(CRT()) = undef

HEAD(ADD(l, v)) = if(LGTH(l) = 0 v else HEAD(l)

LGTH(CRT()) = 0

LGTH(ADD(l, v)) = LGTH(l) + 1

TAIL(CRT()) = CRT()

TAIL(ADD(l, v)) = if(LGTH(l) = 0) (R) diában vessük le

else ADD(TAIL(l), v)

felcserélődött

HF LGTH(TAIL(l)) = ?

? R nem axióma, mert a szintaktikából levezethető

(47) **Petri-hálók**

○ place ——— transition

- csak a másikhoz lehet összekapcsolni

x x x

esetenként lehetnek vajta

- lehet definiálni átfogalmazást: TÜZELÉS

- tud tüzelni, ha van zsetonja
  → input-ból −1
  output-ba +1
- nincs zseton megmaradás !
  - az összes place-ből ki és be a transition különböző oldalára

(48)

P	–	place
T	–	transition
A	–	összekötésekről
M	–	mennyiségről

- egy place lehet in és out is

(49)

M000 ← jelenlegi zsetonkonfigurációja

ha p1 tüzel (2. ábra) – Elérhetőségi – fa

(51) Lefedési fa – hosszú sorban helyezd el azt, ami bármennyi volt, míg egyet.

(52) – meg kell tanulni
Termelő – fogyasztó problémára

(53)
- ki lehet találni speciális tüzelési szabályokat
- lehet tüzelési prioritást bevezetni
- lehet a zsetonra, és pl 10-esre tud tüzelést élesíteni

(54)
- inhalátoros vagy és 1 villa, 1 kanál van! :-)

(0) – lehet beldobni zsetont

2008. 10. 17.

Part 6

(4) Tervezés

- belső struktúrák, részletek kimunkálása
- tervezéssel kapcsolatos döntéseket dokumentálni kell
  - pl.: az ablaknál részletezni kell, hogy mene nyíljon
    - kallizulálni kell az ővel is
- implementálás, tesztelés részletes tervei

(5) Architektúra tervezés:

- nagy komponensek és ezek hogyan vannak összekapcsolva
- az architektúra meghatározza a megvalósítást
  - a felhőkarcoló nem felmagyarázott kutyaól
- adatok biztonságos tárolása, párhuzamosság kezelése, felhasználó kiszolgálása, perzisztencia megvalósítása

Részletes tervezés

- olyan elemi terv, ami alapján a programozónak csak le kell kódolni

(6)
- nem lehet cáfolónak lenni, a követelményre figyelemmel kell lenni
- a döntésnek kivezethetőnek kell lennie a követelményekből
- nem kell feltalálni a kerekeket
- problématerével konzisztensnek kell lennie
- fel kell készülni a változásra
- a vágatlan, kristályosítatlan dolgokon is fel kell kívülni
- nem kodolás
- van a tervezésnek is minősége
- a különböző állapotnak kell lenni, és el kell kéne felismerni

(7) Absztrahció elrejtés / kiemelés

(8) döntéseket hozni a részletek ismeretében

- alkalmazási területtől
- procedurális absztrakció
  - absztrakciót funkcionális finomítással oldjuk
- adat ~
  - bonyolult adatszerkezetet részletesszük
- vezérlés jellegű ~
  - időbeni, dinamikus viselkedéssel kapcsolatos finomítani

(9) encapsulation – egységbezárás

- fizikai, logikai értelemben összetartozó dolgok összezárása

(10) Információ-rejtés

- a döntések mögött álló információk minél inkább el legyenek rejtve egymástól

(11) ~~Modularizálás~~ Modularizálás

- az egyik elem használja a másikat és egymástól függetlenül létrehozhatók (pl. sofőr és fék)
- procedurális absztrakció

* Nobi

(12)
- ~~Egység~~ a kicsi dolgok kevésbé bonyolultak ← létrehozásához kevesebb energiát is kell
- ~~ezt~~ ond meg és vallodj

(13)
- a felül levő modul használja az alsóbbakat

(FC) hallgatók listázása adott átlagnál kisebbek

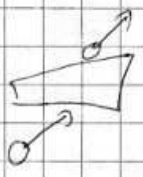
1 ember
- adatokat valahogy el kell érni – adatszerkezet elvése
- le kell kérdezni

2 ember • vezető feladatának – tudni kell, hogy a hallgatói rekord hogy épül fel

3 ember • listázni kell valamilyen formátumban

- M1 főnöknek tűnik – nem ismeri a többi részletet
- ○→ adatszervező ○↗ vezérlésszervező
- nem csak hallgatói rekord adható a getNextRecord()-ra, hanem eof-t is
  - ha rekordot kap adja másnak – átesik a modulon
  - ha eof-t kap akkor leáll – megáll a modul
- vezérlő információ ← ettől megváltozik a modul viselkedése

(14)



Afferens

Efferens

Fan-out – egy kör másikat tud meghajtani, használni (7) a, ó

Fan-in – egy modulnak hány főnöke van
- ha túl nagy – sokan vannak ennek a modulnak kiszolgáltatva
- erre valami ötlet, hogy hány a, ó

- vezérlési hatáskör:
  - a adott modul és általa közvetlen, vagy közvetett módon használt modulok
- döntési hatáskör:
  - csak önmaga és alatti egységek alatta van

(15) Döntéshozatás

- a B-ben hozott döntést meg kell ismételni az A-ban, ha C-t akarják meghívni

(PE) műhelyben emberek vesznek dolgozik, a műhely főnöke és az igazgatóval, csak az akar, ha az igazgató fiát akarják felvenni

- az A és B közötti rejtett megállapodás, hogy ha B szeretné, akkor C-t meg kell hívni
- ha az a rejtett megállapodás le van dokumentálva akkor nincs olyan nagy gond

(11)

2008. 10. 18

ParAG(16-

- tervezés: a specifikált szoftver részleteit fejtjük ki
- döntéseket kell hozni: absztrakció, modularizáció, információrejtés

(16) Coupling = csatolás

- két dolog valahogy egymáshoz van kapcsolva
- függvénynél a paraméterek sorrendje számít - az egyik döntéséből függ hogy mi a sorrend
- A-ból örököl B-t és belőle származik A-t
- a csatolás szükséges, de elkerülendő
- KARBANTARTHATÓSÁG mutatója

(17) 3 dimenziója van:

1. Mi a kommunikáció tárgya, mik az adatok?
2. Mekkora a mélysége?
3. A csatolás milyen mértékű szintaxis valósul meg?

ad (1) adat jellegű csatolás,

- primitív adatot adunk át → jóval kevesebb, de nem felhőtlen élet

stamp típusú

- speci adatszerkezetet adok át – struct-ot pl.

vezérlés típusú csatolás

- másik viselkedését ismerjük, ezért tudjuk vezérelni

közös tárolós csatolás

- közös adatokat kultúr emberi nem használ (pl globális változók)

tartalmi jellegű csatolás

- a másik programját adatlistát bezavarjuk – nagyon ismeri – felépítést, tudja változtatni pl: vírus

ad (3)

- programírás közben csinálják a csatolást – fv-t hívunk
- fordítási időben – ellenőrzi, hogy jól van-e csatolva – fv-t jól hívtuk-e
- linkelési idő – másik modulban van a fv.
- betöltési idő – a címek akkor kapnak értéket
  - a környezeti változók beállítása
- futásidejű – futás közben változó értékek (pl adatbázis kut.
  - oo-nál dinamikus metódushívás
  - ov. hívételek akkor keletkeznek

KOCKÁZATA A HIBÁKNAK!

(18) Kohézió – csatoló fordítottja

- az elem amit kívülről látok akkor mennyi a belső összetartozás.

(19) funkcionális kötődés

- ez a leglazább, nem látunk bele egymásba

pl. sin fv. / egyébként

szekvenciális ~

- bármy dolgot tud

pl: get-valid fv. két dolgot tud

- ha a validot változtatom a get is változhat

kommunikációs ~

- egy adaton, amin bonyolult össze műveleteket

procedurális ~

- pl: printf ← bármit ki tud írni

- beleláthatok, hogy mire mit használ

temporális ~ (idővel összefügg!)

- vannak dolgok, amiket az idő függ össze

logikai ~

- az összetört dolgok között van logikai kapcsolat

- fv - ben sorolni ABC szerint A-F, G-K, K-Z

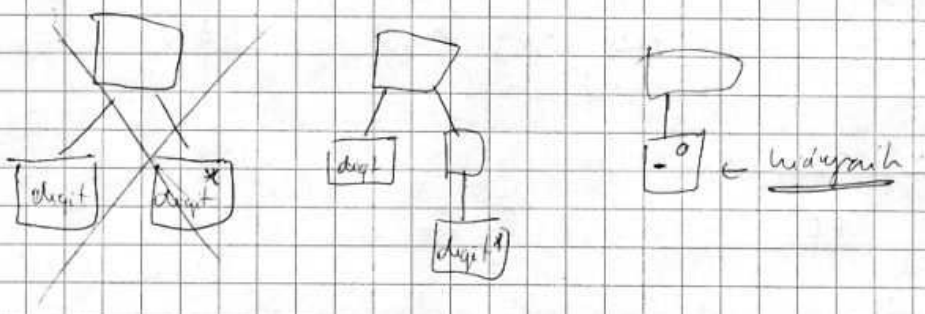
- frameworkök vannak pl kivételek - nagy vérben olyan dolgokat írnak, amiről azt szeretnék, hogy ne fussanak, mert akkor hiba van.

- ezeket is specifikálni kell!!

(20)

(23) Strukturált tervezés – 75P

- Michael Jackson
- ma már nem használják, de a stílust nem vesszük
- adatszerkezet maga után vonhatja a programszerkezetet
- szekvencia: a fenti kilehet van kapcsolva balról - jobbra
- szelekció: X vagy A-ból vagy B-ből áll
- iteráció: X 0 vagy több A-ból áll



(27) Szoftver-architektúra

architektúra típusai:

- strukturálástechnikai komponensek gyűjteménye
- struktúrával és nem funkcióval foglalkozunk

(28) Szoftver arch:

(29)
- hogyan szervezzük össze az elemeket
- egy architektúrához kötött szótár tartozik (templomnál más fogalmak, mint diszkóknál)
- ez is egyfajta nézőpontot jelent

(30)
- rugalmas, egyszerű
- különböző szempontok elválasztása
- felelősségek megfelelő helyre kell elhelyezni

• gazdasági, jogi technológia korlátok

(31) Architekturális ~~###~~ minták

(32)

dir | sort | more — egyik kimenete a másik keresete

felső ↑ ↖ első

(33) filter - transzformál

pipe - összekötők

• sokszor használható, kombinálható

• elégséges szekvenciális / nincs interaktivitás

(34)-(35) Adatbázis kezelő - felépítése

adatkezelési alkalmazások

• jól szeparált az adat és feldolgozás

• az adatbázis konzisztenciája kezelhető - nem tűnik el adat

• nehezen tesztelhető

• hatékonyság érdekében ~~bizt~~ kivétel

• vagy fejlesztői erőfeszítéseket ~~kell~~ kell ráfordítani

(36) control state - állapotlás

(37) OO

• objektumpéldányok vannak és ezek szolgáltatnak másoknak

• jól szimulálják a világ dolgait

• kell ismerniük egymást, hogy együtt tudjanak működni - sorsok - pipeban

(38) Eseményre alapozott

- a történt eseményt mindenkihez eljuttatjuk és abból következtetünk arra valamit

(12)

Part6 (BAAB) 39 -

- rétegelt
- kliens-szerver architektúrák

- Megjelenítő grafikus felület
- Business Object Modell
- Database

- ~~Kliens~~ kliens, hogy nem 3, hanem 2 fizikai rétegben valósul (42 dia)
- fejlődési történet -44. dia
- szolgáltatás orientált architektúra
  - pl: webáruház - biztosításkötés
  - államigazgatásban
  - egészségügyben
  - -> az ~~teszt~~ lesz az irány

WebService (WS)

WSDL -> Webserver Description Language - Webservice leírása

SOAP - Simple Object Access Protokoll
- üzenet fejét, testét definiálja

UDDI - Universal Discovery, Description & Integration
- olyan mint az aranyoldalak

(46) ~~Webszolgáltatás alapfogalmak~~

binding – kötések a címzés módjára

(47) RPC – távoli eljáráshívás

UDDI – siker, ha új piramis épült

(48) SOE – Szolgáltatás-orientált vállalat

BPEL – üzleti folyamat végrehajtó nyelv

- rendszeren belül szolgáltatásokat lehet leírni
- orkesztráció – központi felügyelő fél van
- koreográfia – különböző szereplők kommunikálnak egymással, és így valósul meg a szolgáltatások közötti együttműködés

WS Security

- az azonosítás sok féle képpen lehetséges
  - login-pw, chipkártya, újlenyomatvizsgálat
- jogosultsági rendszer
- a túloldali rendszer is szükséges, hogy igazolja magát valahová
- publikus kulcsú titkosítás
- egy 3 fél hitelesít → ezzel tudod a többiek azonosítani magad

~~WS Policy~~

- valami posta Best Effort
- jó posta – At most once (csak ha tényleg volt, akkor nem egyszer)
- nagyon jó posta – Exactly once ← ezt valaki megvalósítani
- letagadhatatlanság
  - igazolni tudom, hogy pontosan mit küldtem és a vevő nem tudja letagadni hogy ezt megkapta és pontosan ezt!

WS - Transaction

- lehet, hogy bizonyos tranzakciók nem mennek rendesen végbe
  - anyagiasági segítség egy ~~idea~~ helyen vmiért elakad, akkor vissza kell csinálni a dolgokat - ROLL-BACK

Az egész SOE-val az a gond, hogy platformfüggő

- egy cégen belül nem gond
- de pl egy államigazgatásban több-féle rendszer van, melyek ezeket összehangolni

Part (7)

FSD

- Fachlich System Development

(Pe) csinálódó tárgy szoftver

- vagy hány csinálódózás volt, mennyi ideig tart egy átlagos csinálódózás

- 2 gomb, start csinálódózás, stop csinálódózás

(4) A megvalósítás:

- ebből nem lehet leghosszabb csinálódózási időt számolni ← 1 hónap múlva ezt hívni fölösleges
- ebben nincs benne a csinálódózás tényleges, mert a funkcionalitás volt az első dél
- a modell vonzására hálózatszámítás a funkcionalitást
- olyan projekt kell ami a valóságot szimulálja

- a világot össze kell kötni a szimulátorral
- entitások: történések szereplői és objektek

(6) Bankot nyitunk

- egy embernek vezet számlát
- invest - bankszámla megnyitása
- pay-in befizet
- withdraw - kifizetés
- termination - kilép az ügyfél

- Legyen két ügyfél
  - csak egy új gép + 1 új ügyintéző kell

- Legyen 40000 ügyfél :-)

(13) 2008. 10. 29.

Lec 9-re lesz kijavítva

(HF) nov 3 - nov 25 kedd → addig kell Fenni pogy előadás szünetében is lehet

- 6db példa, 1 példát még csak jövő hét után lehet megoldani

  4×8 + 2×4

- példánként fele pontszámot el kell érni
- kézzel kell megoldani és papíron kell beadni

Part II - 7 -

ER - lépések

- entitáshalmaz - lépés
  - kell az entitáshoz vagy kell valami lépések vannak

- entitások élettörténete

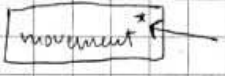
- kiinduló modell
  - élettörténetből processt csinálunk!

- funkciókat adunk a modellhez
- időzítési lépés – modellezésre, funkcióhoz vonatkozó időbeli korlátok

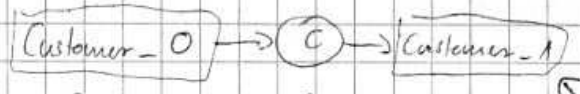
(9) - bankot kell csinálni
  - bankszámlát nyitni tudunk – 1db
  - tranzakciókat végezteni kell – 2 funkció
    1, ha az ügyfél számlára megy, akkor jelezni kell
    2, lekérdezhetjük az ügyfél egyenlegét

(10) Entitás: ügyfél (bankszámla)

Akciók: - számlanyitás
- ki } fizetés
- be }
- számlazárás

(11) 

1-nek, vagy többször fizethet ki-ba

(12) 

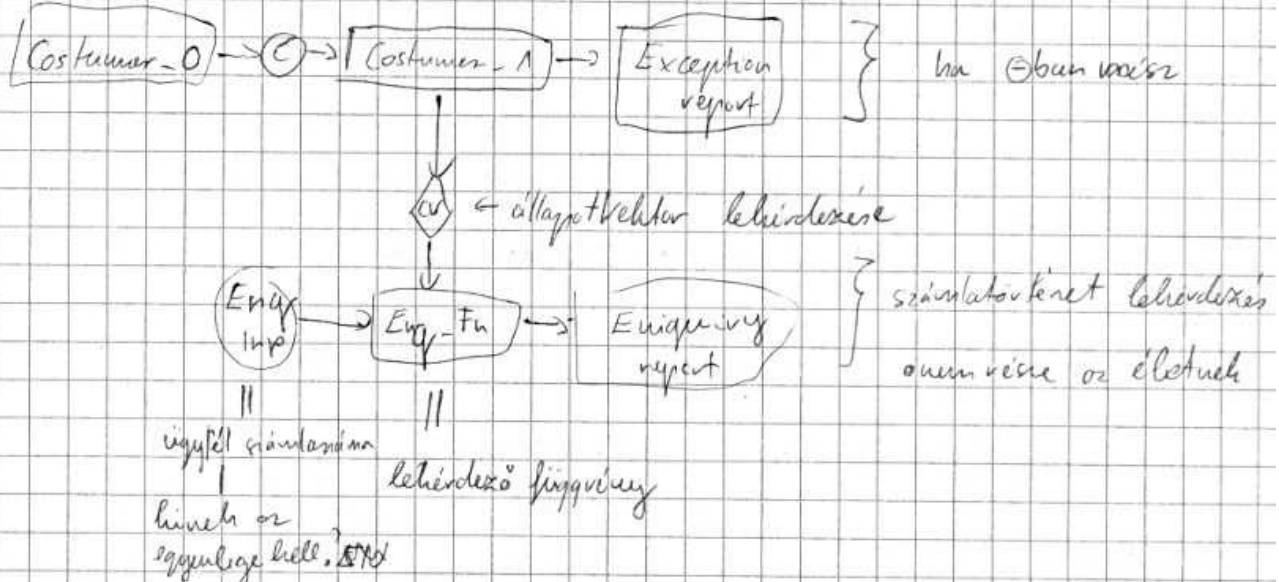
a modellben végbemenő folyamat

fizikai világban végbemenő folyamat

üzenetsor

- indító lépésnél csak „invest" kell az üzenetsor
- később pay-in, withdraw vagy termin

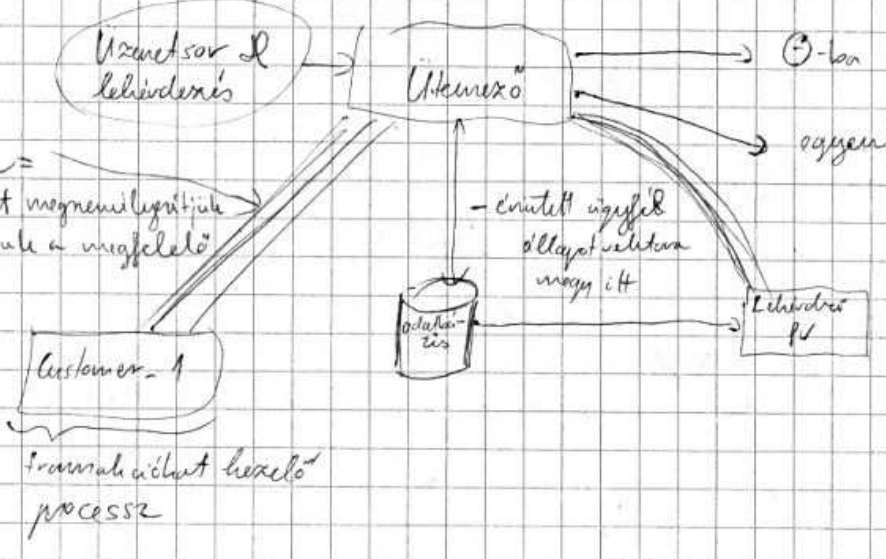
(13) Funkciók



(15) Időzítés

- időbeli feltételek bevezetése
  - ha ⊖-ba megy – azonnali beavatkozás / záráskor nézzük, hogy ⊖-ban van-e
    - időzítésnek nincs része az entitás leírásában és a funkciókban sem
  - mikor lépjen vállalat az egyenleg – azonnal / záráskor
  - mi az egyenleg tartalma – aktuális / utolsó záráskor / ...

(16)



(18) Napvége - naplezdet ütemező

first part

↓

- előző napi egyenleg
- élettörténet eseményeke záródnak távolódnak

second part - napvégehez

- élettörténet eseményeket keresnek

(19) Entitás alkű lépés

Entitások

- fel kell ismerni hogy az entitásba, milyen alkűk fordulnak elő
- a főnevekből lesz entitás
- alkalmazási területről jönnek
- alkűkat definiálnak személyek el
- típus - például gondolatvilágban kell gondolkodni

Alkűk

- igék, igenevek
- cselekvések
- a külső világ a forrása, vezérlője az alkűknak

(20) ELH - entitás életútja

- egy esemény több entitás életére hathat

Olvasó - belép / kölcsönöz, visszahoz / kilép

↓

ütemes szekvencia, ciklus szelekció

(21) Könyv - beszerzés / kiadás, visszavétel / kiselejtezés

↓

szekvencia

(22) - egy-egy üzenetsor végtelen FIFO-nak tekinthető

- olvasni (csak akkor, ha nem üres) / írni mindig lehet!

Bal ábra

- két küldőtöröl helyes olvassa C_1, C_2-t, a tartalom nem keveredik

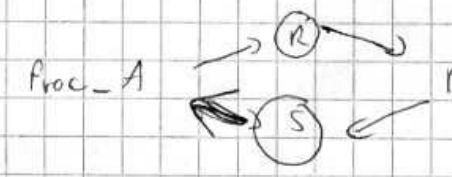
Jobb

- a kettő közül (C_1, C_2) az egyikből olvas

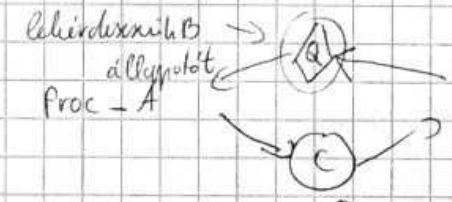
(23) Q

P-proc olvassa P állapotvektorát / akkor amikor P állapota jól rendelkezésre áll.

(24)



• B-nek küldök üzenetet, és várom B válaszát



• az üzenetet fogadó processz, alkalmas-e a fogadásra

(25)

- input processz - a bemenő adatokat szűri, és azokat továbbítja a model process-nek

→ vagy lekérdező - vezérlés

→ iteratív fv - modell változtatása

(26) a, funkciói a modellben

b, állapot-vektort lekérdező funkciói

(27) Interaktív funkció – kamat számolás

(28) Időzítési lépés

- állapotvektor olvasás milyen gyakran megy végbe?
- 10-szeres mintavételezés kell!!
- óvás, Linearek állhatnak, amik eseményeket generálnak — pl eljön az ideje, hogy felhelyjünk az olvasásból

(29)

Inversion

- Q mutat először (P a Q vége), vA ives, hívjuk P-t
- a P is eljárásként hívja Q-t és visszat

(30)

2008. 10. 31.

(14)

Gyakorlás

(1)

Egy adatfolyam -ában szereplő vezérlőegységet az alábbi állapotgéppel írjuk le.

Vezérlő egység ~~bemenetei~~ bemenetei: béta, X, c3

processek: alfa, ubul, X2 ÁLLAPOT / folyamat

adatfolyam: az nem derül ki ebből

(2) Konstrukciók: CRT INS

behavior HBR, SIZE

modifier: RMV

SIZE(CRT()) = 0

SIZE(INS(s, i)) =

- halmaz → ha vagy ezt megjegyzem valahol, akkor nem változik a méret
- max 10 elem

= if(size(s) < 10 && !MBR(s, i)) size(s)+1
else size(s)

MBR(CRT(), i) = F

MBR(INS(s, i), j) = size(s) < 10 && i == j || MBR(s, j)

i nem kell ide, mert boolean a visszatérés, nem halmaz

ha még bele lehet valami és i == j, vagy már benne volt

RMV(CRT(), i) = CRT()

RMV(INS(s, i), j) =

if(size(s) < 10 && i != j) INS(RMV(s, j), i)

else ~~if~~ (~~size(s) < 10 &&~~ i == j) · RMV(s, j)

ez nem is kell, mert erről nem függ !!

- annak nincs jelentősége, hogy i-benne van e a halmazban

Konstruktor: olyan művelet, ami kell az összes létező szekvenciához

③ Konstruktorok: CRT, ADD

Behaviorok: END, DUPLO, LAST

Modifierek: —

LAST(CRT()) = undefined

LAST(ADD(s, x)) = x

END(s, CRT()) = T (ilyen sztringhez akarok hozzá !!)

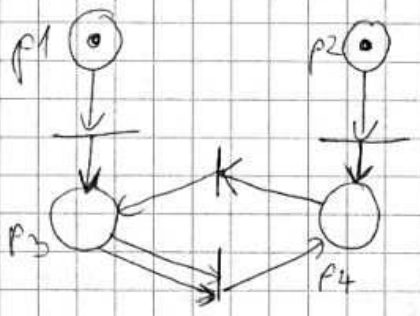
END(CRT(), ADD(s, x)) = F s-hez s végén áll, és azt akarjuk hozzá

END(ADD(s1, x), ADD(s2, y)) = end(s1, s2) && y == x

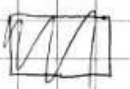
DUPLO(CRT()) = F

DUPLO(ADD(s, x)) = duplo(s) || last(s) == x

(4)



(5) Fadiagon ábrát! DTD-ből

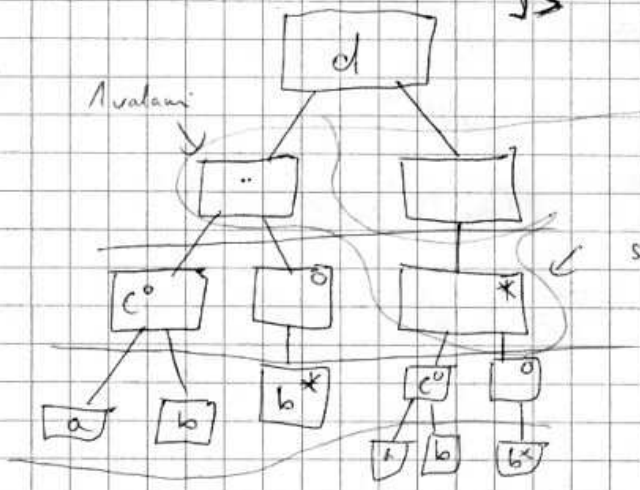


<!DOCTYPE d [

<!ELEMENT a (#PCDATA) >
<!ELEMENT b (#PCDATA) >
<!ELEMENT c (a,b|) >
<!ELEMENT d (c|b+|⊕) >

egy vagy több izéből áll

]>



Objektum_orientáltság

(1) Általános bevezetés

(2) UML

(3) RUP

(3) Az OO története

- Strukturált programozás – 70-es évek eleje
  - program = absztrakt gépek lánca
- előnyök: világos struktúra
- (4) - problémák: szűk: változók, globális változók, eljárások duplikálása
  - láthatóság és fa-struktúra egymásnak ellentmondanak – fa alsó ágain fv-t hívni ⇒ 2 ágban
  - fragmentálás megoldhatatlan

(5) Moduláris programozás

- pl: C
- nagy programot elkülönülő darabokra a lexikális elv alapján

(6) Absztrakt adatstruktúrák

- a műveletekkel és ne az implementációval foglalkozunk

(7) Előnyök – lásd: javasolt megoldások

- de probléma volt a példányosítás
  - a struktúrában belső állapotjelző kellenek ⇐ az eddig rejtett struktúra publikussá válik
  - vagy copy-paste
- infix operációk – 3+5

  prefix – összead (3,5)

(8) OO paradigmák

- van objektum, ami viselkedik, működik

(9) Objektum fogalma:

- objektum az ami szolgáltat, és felelős egy bizonyos szolgáltatás kimenetéért
- az objektum kliense veszi igénybe a szolgáltatást (a kliens objektum is lehet)
- objektum: azonosítható, egységbe zárt entitás, ami szolgáltat

(10) Két objektum: file – printer

- kinek milyen a felelőssége? „ki mért felel?"
- a nyomtató felelőssége a nyomtatás
  - a nyomtató ismeri az összes lehető fájltípust? NEM
- a fájlnak az a felelőssége, hogy átalakítsa magát olyan formára, amit a nyomtató ki tud nyomtatni (PCL, PostScript)

(11) Kérés

- ezzel kell kérni a szolgáltatást
- milyen érvégzet egy szolgáltatás, milyen típusú jelzőkészülékek lehetnek
  - a, lehet pl várni arra, hogy kiszolgáljanak
  - b, ha a vonat meg üti a bokát, akkor az exceptiont.
  - c, elmegy a tisztítóba, átkapcsol a kábeltől és visszajössz épte
  - d, egyéni asszisztenciás kérés – vonat ki az ablakon és lóton kalapot és kivod a mentőt – nem kivod vissza, és nem kiszáll vissza → NINCS visszacsatolás !!

2008.11.05. (15)

(11) • objektum az ami szolgáltat
  - ez felelősséggel jár
- a világ: objektumok együttes interakciója
- request: szolgáltatás kérés
  - a kérés feltételezi, hogy van valakinek a kérő és kiszolgáló között
  - a kérés kezdete egyértelmű, vége nem
    - lehet, hogy meghalnak – lagymatag esetek
    - macskát meghajmálnak kivételes helyzet lép fel – valaki más meghal
    - elindítom a kérést, és csak később kérdezd az eredményt – tárolás

• csak indítod a lekérést / eredménye nem érdekel – nemtőlekérés

(12) <u>információk, hogy lekérni tudjanak:</u>

• mit akarnak kérni

• mit kell felkérni – tisztítókból nincs pizza

• paraméterek

• ~~lekérési~~ lekérési tartalmak – pl: pizzákat

<u>Kérési űrlap</u>

• formalizáltságot növeli

• <u>ürtéke</u>: form egy mezőjében elfogadható adat

• objektum referencia: ezen keresztül lehet elérni az objektumot

pl: referencia, rendelésszám, cím, pároz

• egy objektumra több referencia is hivatkozhat

(13) • <u>a kérés</u> = teljes nevében végrehajtott tevékenység

• milyen jogosítványokkal rendelkezik a szolgáltatáskérő

• speciális esetekben kivételek lépnek fel

(14) objektumok elpusztítása, törlése során kérésekkel ~~a~~ vajthető végre

<u>típus</u>: pl integer – predikátumnak felel meg

<u>kiterjedés</u> (extenzió) – azok elek megfelelnek a típusnak

(15) <u>objektumtípus</u>: elemei objektum referenciák

(16) <u>interfész</u>: deklarációja valamiféle összetartozás ~~jellemzőjének~~ jellemzőjének

• szerződéses pontokat definiál, hogy hogyan lehet az adott szolgáltatást, kötelezettséget megvalósítani

<u>típusa</u>: objektumtípus

(17) ugyanaz kell lennie a Liskov féle helyettesítésnek

A compatible B, A is-a B – helyettesíthető

egymással

(18) reflexív, nem-szimmetrikus, tranzitív

(19) operáció: valamilyen értelmezhető szolgáltatást jelent meg

- lekérés hatására operációnak kell végbemennie

[Guarany] <visszatérési típus> <azonosító> (param 1, ... / [kivételek] [tartalom]

↖ egyszerű – mint a mentésnél

(20) Paraméterek:

- módja: ~ átadás iránya (in – felülről adja szervernek
  out – szerver felülről
  inout – mindkettő lehet)
- return – speciális out paraméter

(21) végrehajtási szemantika

- at most once: legfeljebb egyszer – ellenőrzés példa amikor lényeges
- exactly once: pontosan 1x ← lekérés megvalósításai
- best effort: mentős példa

(22-25) Implementálás (-21-)

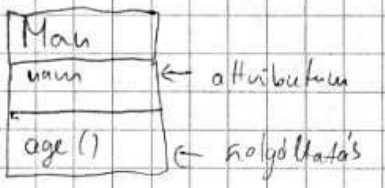
- végrehajtási modell: valami – gépi kódot kell végrehajtani – METÓDUS
  - a metódus meghívása ⇐ metódus aktiválása = folyamat indítása
- konstrukciós modell: objektum belső váltózóle
  - létrehozni kell úgy, hogy mi volt előbb – pl
    hogyan lesznek tőle szolgáltatást

26 Konstrukciós modell

- viselkedés: szolgáltatáshalmaz amit nyújt
- belső struktúra: arra alkalmas kell, hogy meg lehessen implementálni
- állapot: struktúra kitöltöttsége
- identitás (egyediség): egyediséget biztosítja

27 Osztály

- definiálja a hasonló struktúrájú és viselkedésű objektumokat
- gyár - objektumminta

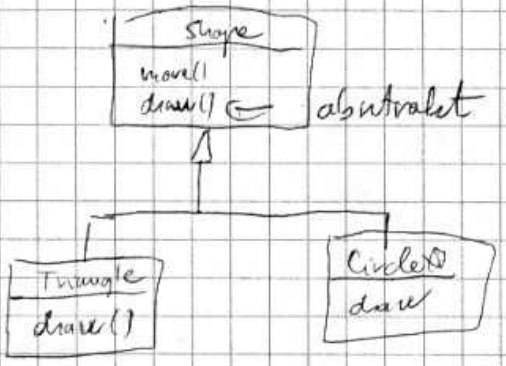


28 Öröklés (inheritance)

29 Subclass - leszármazott osztály

- nemcsak kiegészít, hanem módosít is - átdefiniál

30 Absztrakt műveletek



31 Szerződés alapú együttműködés

- jöjjön létre megállapodás az objektumok között

(32)

• Különböző műveletek egymás hatása után és változatlan

pl: bankon belül saját számlájáról saját számlára ~~nem~~ nem lapjat

CLASS ~~HAVA~~ INVARIANT

(33) Eiffel-nyelv

(PE) bankszámla – csak betenni lehet rá (deposit)

• nézi az előző működéseket

require – precondition (előfeltétel)

ensure – postcondition (működéstől)

invariant: két befizetés között nem történik el pénz (csak betenni lehet)

(35) Öröklés

• osztály öröklés: csak arra használjuk, hogy egy másikat kiterjesszünk belőle

• interfészöröklés: igaz-e, hogy a Cendrmorott az és helyére behelyettesíthető

(36) Kör – ellipszis probléma

• ezek nem származtathatóak egymásból

(37) Nyitott – Zárt elv

• nyitottnak kell lennie a bővítésre
• zártnak a módosításra, felülírásra

} – osztálynak ilyennek kell lennie

(38)

(39) Readline nem Line

Öröklés: az autó motorja nem egy autó

(40) Öröklés és a szerződések

r' - ez ugyanaz, mint r, csak A'-ban

(41) Def: p erősebb q, ha p minden autón van q-t

- invariáns erősebb, vagy egyenlő kell legyen, mint inv
- redukálhatósági szabály: a leszármazott előfeltételének enyhébbnek, az utófeltételének szigorúbbnak kell lennie

(43)

- ha alul r'(B b) helyett r'(B' b) nem jó!!

```
A[] myArray = new A[10];          // A'[10];
myArray[5] = new A();  // new A'();   // A();
```

(16) 2008. 11.07.

(43)

(44) Objektumorientáltság

- a változó egy zseb, amibe bele lehet tenni dolgokat
- milyen objektumot lehet beletenni — statikus típusozás — fordítási időben tudjuk, hogy milyen típust lehet beletenni
  - dinamikus típusozás — bármi mehet bele
- ki felelős a zsebbe tett ~~dolgokért~~ dolgon végzett, műveletért?
  - a zseb - statikus binding
  - akit belerakunk - dynamic binding

(45)

Típuszálás

kötés	statikus	dinamikus
statikus	NEM OO	??? - a szélsőbe valami és ellen művelet
dinamikus	C++ ha gyenge típuszálás van ⇓ a valós kompatibilis típusokat is elfogadjuk	Excel alla ⇓ mennet közben nincsenek kapcsolatok

(46) Demeter törvénye: ne fogadj el ordinálódót a adatosbázistól = ne állj szóba idegennel

OO világban lesz az ősöcskéből?

- legyen C osztály M működéssel
- M paramétere, ~~this~~ magamat, ősömet, a C osztály attribútumai, átmeneti változók (ciklusváltozó), globális változók / olyan objektumok, amelyek az osztály gyárt ⊂ ezek a közeli barátok

(47)

- a fordító megeszi
- De a-ból nem lehet tudni, hogy van B, C tartalma
- ha megváltozik B struktúrája, azt lehetne kellene A-nak

(48) Ne csinálj felesleges garanciaszerződést, ha nem muszáj!!

Part 9

UML

- OMG - object management group - az objektumvilágban alkalmazandó szabványokat csinál pl UML-szabvány

www.omg.org

Unified Modelling Language - modellező nyelv

- két része
  - vizualizálás
  - OCL - object constraint language

(3) - egy modellező nyelv - vizualizálás, specifikálás, construálás, dokumentálás

- nagy és bonyolult rendszerek leírására alkalmas, nem csak software-rendszer

(4) (5) - általános keretet definiál, amit bővíteni lehet

- modellező nyelv - nem a folyamatra vonatkozik
- kód nagy része előállítható a diagramokból (generálni lehet)

(6) - nem programnyelv

- nem tool
- olyan modell leggyen, ami általánosan elfogadott standard - ha a tool megszűnik, akkor simán másik toolra át lehessen vinni
- nem definiál folyamatot

(7)

- nagy: infrastruktúra
- superstruktúra ≈ UML - infrastruktúra
- [illegible]

(8) az UML is UML-ben van definiálva

----> függőségek jelölése

(9) gráf = UML diagram

- vizuális reláció - kapcsolat
  tartalmazás
  vizuális hierarchia jelentősége
- ikonok vannak fix méretben és alakban
  - útvonalak - egyenes vonalak

(10) elnevezés: scope-on, package-en belül egyedi ← lokális

címke: grafikus elemnek adjunk nevet ← nem feltétlen egyedi

megjegyzés:

(11) 3 db általános kiterjesztő mechanizmus

- ~~kiterjesztés~~ korlátozás add(1) ------ { ordered } → vsegy rendezetten adjahozzá
- sztereotípia << exception >> pl: interfész
  - a dolgokat megjelölgjük vele
  - jól kiemelhetők a kódgenerálásban — kompájlálási művelet kapcsolható hozzá
- tagged value { version = 3.2 } — név, értéke pár
  - jól kiemelhető a kódgenerálásban

(12)

(14) Class-diagram

- osztályokat ábrázolunk le vele
- többféle részletességi szint lehetséges
- név — attribútum — metódus

↑

ha absztrakt, akkor dőlt a neve

(15) Név dolog

- név
- sztereotípia — interfész, utility, metaclass
- tagged value
- lehet illust is vagy leír

metaclass

pl: ~~Class~~: azon osztályok, amelyek példányai osztályok

pl: utility: olyan osztály melynek attribútumai, operációi deklaratívak

pl: cos lr — semmi nyoma nincs a példányok működésében
— sokan használják

(16) attribútumok

name: String – név – típus pár

~~-~~ height : Integer = 5

/ – őstől örökölt attribútum

(17) Láthatóság

- public +
- protected #
- private -
- package ~

- ha nincs jel ⇒ akkor nem jelöltük

(18) ~~A~~ ontológiaváltás – nem a példányé, hanem az ontológié

a: int = 0

a1.a = a2.a + 3 ~~= 3~~ a2.a = 3
a2.a += 4 ; a2.a = a1.a = 7

A.a = a1.a - 2 ; a2.a = 5

(19) előírásosítás

- readOnly
- subset – alhalmas részhalmaza property-nak nem
- union
- redefine
- ordered
- unique
- :

(20)

rendezett	egyedi	
-	+	halmaz
+	+	rendezett halmaz
-	-	csomag
+	-	sorozat

(21) enumeráció

pl: évszakok

<< enumeration >>
Évszak
Tavasz
nyár
tél
ősz

(22) Ha osztályban osztálya mutatunk

- A -ban referencia B-re ⇐ asszociáció egy megvalósítása

(23)

2008. 11. 12

(17)

2009

Part 3

(17)

(24)

query - nem változtatja az objektum állapotát

redefiníció

ordered, unique - egyedi, rendezett érték

(25)

class - scope - osztályszintű - statikus

(26) Operációk

- szekvenciális: sorrendi - kiírja a konkurencia-problémát
- guarded: piszkáért jönnek mielőtt a másikat befolyásolná volna - ironagyát kell valami a piszkálás elé!!
  - így se fognak egyszerre több piszkát csinálni
- valódi konkurencia: valahogy uralni kell a konkurens igényeket

(27) Aktív objektum

- akkor is szolgáltat, ha senki sem kérte - pl main fv.

B

(28) Template

- paraméterezni lehet és így állítunk elő típusokat

(29) CLASSIFIER

• osztály-szerű dolgok, amik hő úgy viselkednek, mint osztály

(30) Relációk

• függőség – referencián keresztül x.doIt()-t hívni

↑ az függ a Stack-től

• pl: csekkolás bőről és vevő lépőtt függőség
fürj felelős kiállít más

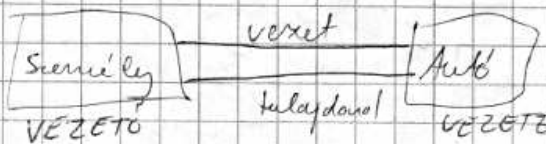
(31) Asszociáció: objektumpárok kapcsolatainak absztrakciója

• semantikus kapcsolat 2 dolog között – erősebb mint a függőség

• hosszú távon ismer egy objektum egy másikat

• pl: stabil attribútum – letárolt objektumreferenciája

↑ Későbbiekben is emlékezik rá az objektum



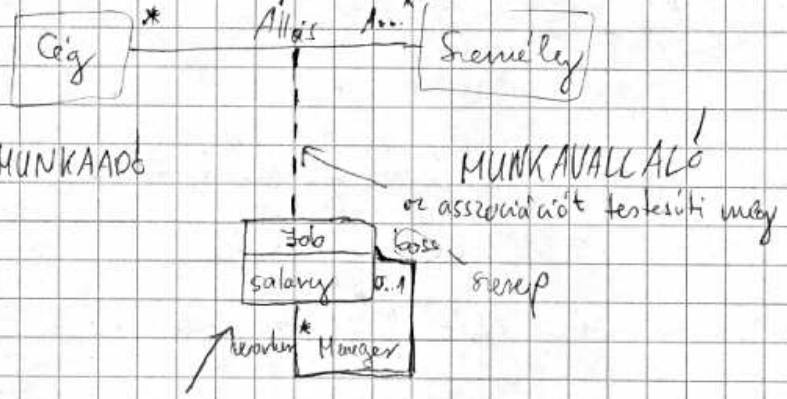
↖ szerep: ezek lesznek a kimenő referenciák

• az asszociáció példánya a LINK

• asszociáció nem biztos, hogy fordítva is felírható

• két végéhez szerepet adhatunk rendelni

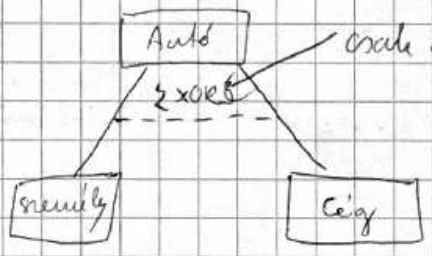
(32)



Kerset-referencia-tábla

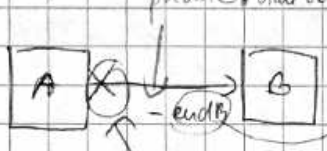
• ha a fizetés a személyben, vagy a cégnél megy, az nem jó mert akkor valami változás-adatmozgatást kína kelti

(33) / - leszármaztatott asszociáció



(34) Navigálhatóság

- melyik irányból járható át az asszociáció hatékonyan



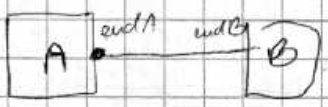
- A-ból B-be hatékonyan lehet átmenni

nem navigálható: B-ből hatékonyan nem érhető el A.

- az plus egyszer hatékonyan elérhető a felhasználóiról a jelszó

(35) ownership: tulajdonlás — ki birtokolja az asszociáció-véget

- ehhez ül a végén



(36) Minősítők

- ha lehet kerülni

(37-39) Aggregáció

- rész-egész viszony kifejezésére

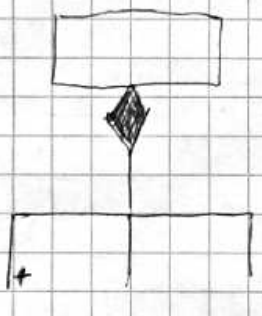
- kompozíció

- az aggregáció erős változata + hosszú-fejezet → tartalmazás — ...
- a hosszú megszüntetését a tartónál kell kezelni
- a hosszú karbantartásához a tartalmakat kell karbantartani, valamint a fedőlapot

• deep – shallow copy
↑ ↖ csak a címlapot másolják
lemásolják és másolják az egész szerkezetet

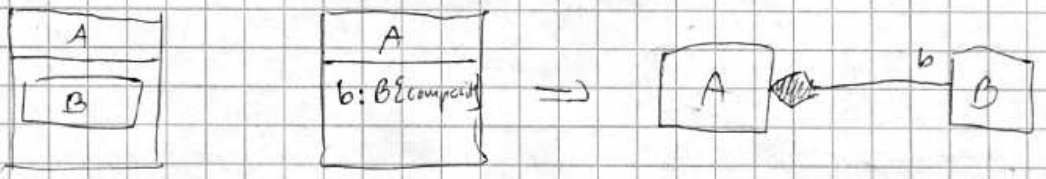
• osztott aggregáció
  • gyengébb aggregáció
  • egy komponens nem kizárólag egy elembe tartozik
    pl: szobák falai több szobához is tartozik
    ⇓
    osztoznak a falon

(39)



• a tele – komponens aggregáció
• az üres – osztott aggregáció

(40) Bedugozott osztályok



(41) Generalizálás – öröklés

(42)
• egy dimenzióban lehet valami
  • a nemek szerinti öröklés nincs egy dimenzióban a munkavállalóval

(43) Party: egy kollekciója a lehetséges osztályoknak

(44) Többszörös öröklés

(45) Interfész

- operációk neve + elérhető szolgáltatások és hiteleshetőség
  koherens szolgáltatásai
- nem példányosítható

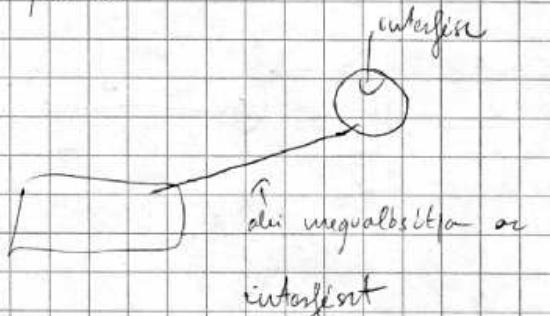
(46)

- 2 fajta van
  - amit példányosítani – otthon: apa interfész, egyetemen tanár
    ↑ provided
  - elvárt – valakivel együtt kell működni, aki elvárt módon működik
    ↑ required

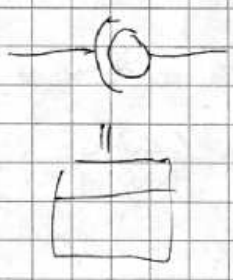
(47) <<interface>> ← sztereotípia

- MyClass – függőség
- MyString – ⇢ – megvalósít / realizál, implementálja
  ⇢ ← végi jelölés

(48) Nyalóka jelölés



(48) Elvárt interfész



(49)

(50) Objektum

Jim : Person — jim a person példánya

Jim

:Person — Person osztály még nem nevesített példánya

(51)

- a konkrét osztályok a UML class példányai
- Instance Specification — META-elem

(52) Link

- konkrét objektumpéldányok közötti kapcsolatok

officer — szerep

(18)

2008. 11. 14.

Gyakorlat

(2) Milyen UML elemek?

- aggregáció — ◇ — nincs
- függőség - - -> — nincs
- öröklés metódus — alábírott metódus — van
- multiobject — ⇐ nem található
- realizálás - - -▷ — van
- kollaboráció
- navigáció ——> — van
- példányosítás — nincs

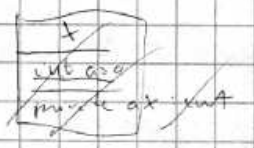
- qualifier – minus
- absztrakt osztály – minus ⊂ dőlt betűs a neve
- sztereotípus – << ... >>

- ha példányosítás / akkor a példányosítandó osztály függ attól, amiből példányosítás

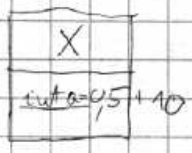
(3)

x2.a = x1.a + 5

x1.a += 5



end x1.a = 10 x2.a = 10



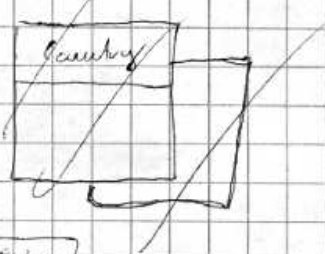
x1 : X x2 : X

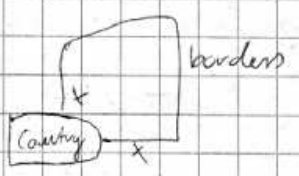
(4) Objektumdiagramból → osztálydiagram

↑

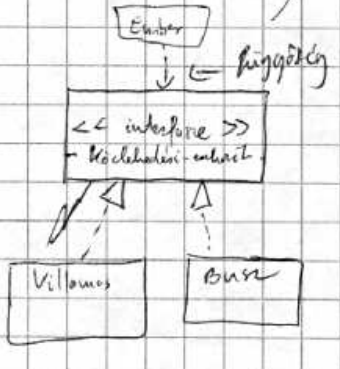
példányosítási jelvétel

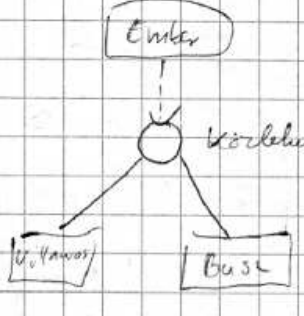
- Country osztálynak 3 példánya és határaik közös asszociáció





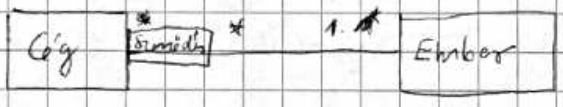
(5)





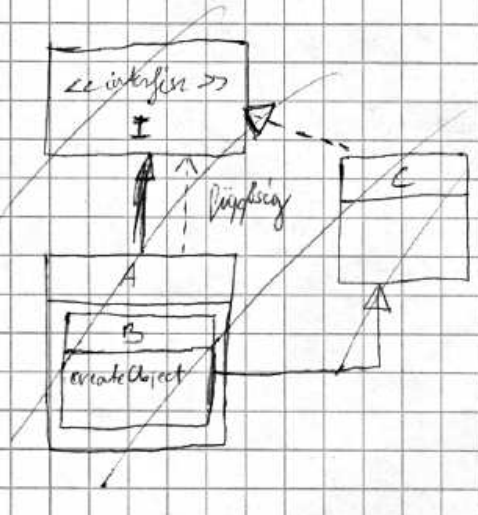
(6)

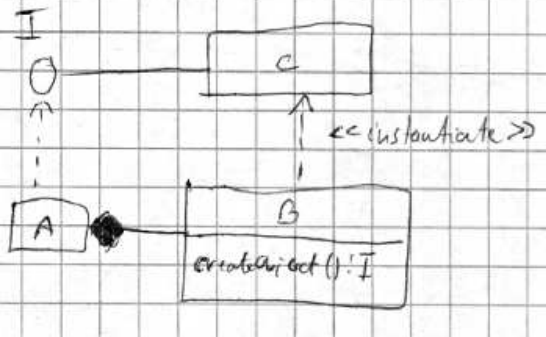
qualifier: attribútum egy halmaza, ami leszűkíthető a multiplicitást



• 1 cég-nek egy nevűdése van egy cégnél

(7)





• A használni akar, de nem akar példányosítani

(8)

A/4

[A] [D] B bárhol helyettesíthető E-vel, mert E a B leszármazottja

[E] C bárhol helyettesíthető D-vel

[D] lehet, mert A interfészt — D megvalósítja A interfészt

[B] Kaphat paramétrül E-t, de nem hívhatja, mert B-nek nincs count-ja!! ~~A Greet nincs count-ja~~

[C] E-ben van asszociáció D-re → E ismeri D-t

[A] D-nek nincs bar metódusa → mert private volt

[B] F nem látja D-t

[E] add() látja do-t, de nem absztrakt

(16) [B] – egy példány valamely

[D] – lapelhat R-t!!

[A]

[P] – lehetővé tett és aggregációs kapcsolat Album felé ⇐ ez az index

[G] – megvalósítás miatt van asszociáció

[C] – asszociáció miatt létrehozhat R-t!!

• ismeri Q-t, hisz interfészen törölhető a kapcsolat

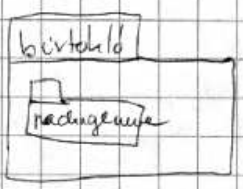
[B]

[E]

Part 9 (53-)

(54) Package – csomagolt szerkezet

• névteret jelöl ki – egyedi névtérnek kell lennie



⊕ ← tartalmazás

(55) Láthatóság

• public | private

• kapcsolás

• <<access>>

• <<import>> – látható a cs, ami többi példányban van

Webshop-ból látjuk Types-t is! de

(56) Tartalmi összefoglalás

<<vége>>

(57) • jobb oldal amiből → ami balon az eredmény lesz!!

(58) Komponens diagramm

Komponens: olyan fizikailag létező elem, ami interfészeket realizál

• van fizikai életciklusa

• Típusai

• deployment – szükséges ahhoz, hogy a software telepítés után elinduljon

• adatfájlok, konfigurációs fájlok

• munkatermékek: a fejlesztés során, tesztelés során létrejött produktumok

• végrehajtással kapcsolatos termékek

• futás időben keletkezett fájlok, peldanyositas objektumpéldányok

2005. 11. 26

(85) Integrációs diagrammok – 4 fajta

(86) Szekvenciadiagram

• időben hogyan reagnak ki az együttműködéssel kapcsolatos dolgok

• kölcsönhatásokra sorrendiség

Kommunikációs diagram

• szerkezetre épül – szerkezeti összefüggéseket a kommunikáció felett bírt

Interakciós áttekintő diagram

Timing Diagram

(87) Interakció: különböző elemek együtt működnek és ez a együtt működés lépéseiből áll

Trace (nyomkövetés): történések sorozata, eseményes → hatást gyakorol

(88) Események: • tevékenység megkezdése / zárása
• objektum létrehozása, megsemmisítése
• üzenet küldése

Üzenet: két-működés specifikálása, melynek célja szolgáltatás elvégzése, tevékenységet old a fogadónak

Jelzés (signál): a jelzést fogadó objektum számára jelen, tevékenységre nem számítunk, valami nem várunk

Send Operation Event — küldés
Receive Operation Event — fogadás
eseményei

SendSignal - ReceiveSignal

(89) Interakció: eseménysorok két halmaza. Érvényes eseménysorok, érvénytelen eseménysorok

• ha kompenzálják, akkor zárt marad
• ha átlapolják / az eseménysorok megmaradnak

(93)

Lifeline - az idő lefelé múlik
• ez üzen a-nak (oper 1)
• ezt az tevékenységet - oszlop jelenik meg (execution specification)

küldött üzenet ↓ →

• aa visszaállítja C-t (ha kell neki valami callbackhez)

• amíg aa dolgozik, addig cc vár

• cc kiszolgálja aa-t – callbackhez tartozó tevékenységet végez

• ha végzett ---> c er a vetem

• és aa is vetem --->

• előfordulhat, hogy aa nemcsak cc-vel, hanem C egy másik példányával is kommunikálhat

• az ábra – <u>asztalosok közötti együttműködés</u>

(94)

• execution specificationt nem kell minden fölrajzolni, de ajánlatos

• ——> aszinkron üzenet

• végi példában más volt a jelölés ——> szinkron

——> aszinkron

• aszinkron üzenetnél ~~kell~~ lehetséges is egymást a nyilak, csak felfelé nem mutathat nyíl! (pl. karbantartója később kerül kiadásra, mint a feladat)

• saját kerek sérülve lehet üzenni, és más ide tud kapcsolódni

(95)

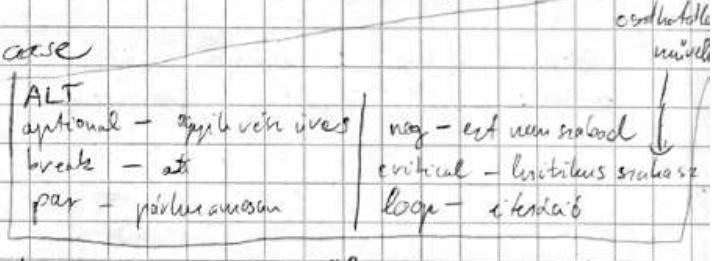
• operátor hatására indul a folyamat – kívülről indítja valaki

• frame a frame-ben + szerkezet - case

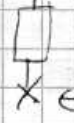
régiek

} ALTERNATE

ALT	
optional – egyik vétel üres	neg – ezt nem szabad
break – ezt	critical – kritikus szakasz
par – párhuzamosan	loop – ciklusok



– create –> :C2 – új osztályból példányt csinálunk C2-ből régen: 



X ∈ megdöglött a példány

• mindenhol követi a vámutató referenciát – destruktor = utolsó lecke

(96) rej – rejtély

(98) Kommunikációs diagram

- együttműködők

Kollaboráció: vezérlés 1 funkcionalitást /...

(99)
- nyílak nélkül olyan, mint egy class diagram asszociációk
- üzenetekből kirajzolódik egymásutánja – szekvencia diagrammal is megvalósítható
- az üzenetekhez nyilak ábrázolják → szinkron
  ⇢ aszinkron

1 - első üzenet
1.1
1.1.1

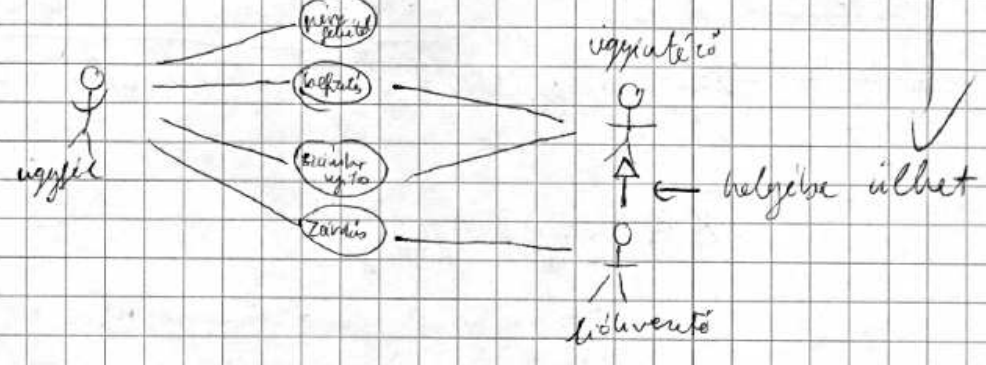
2
2.1 - 2.8 – loopol megadva
2.1 : [i=1:8] ms(i)

Gyula

zárás kérvényező

(A) Egy banki rendszerben bankkártyával pénzt tudunk felvenni egy számlától, banki ügyintéző közvetítésével lehet számlát nyitni és befizetni. A számlát zárolni csak a fiók-vezető tudja, neki kell rendelkezni az ügyintézési jogokkal. Uml use-case diagram!

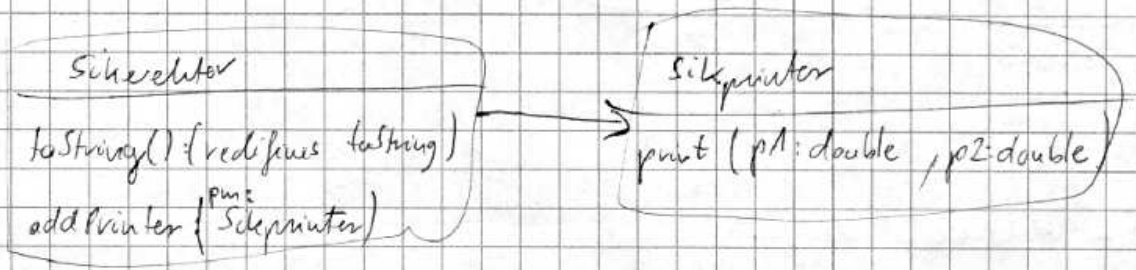
use-case – ige: felvenni, nyitni, befizetni, zárolni
actor: ügyfél, ügyintéző, fiókvezető

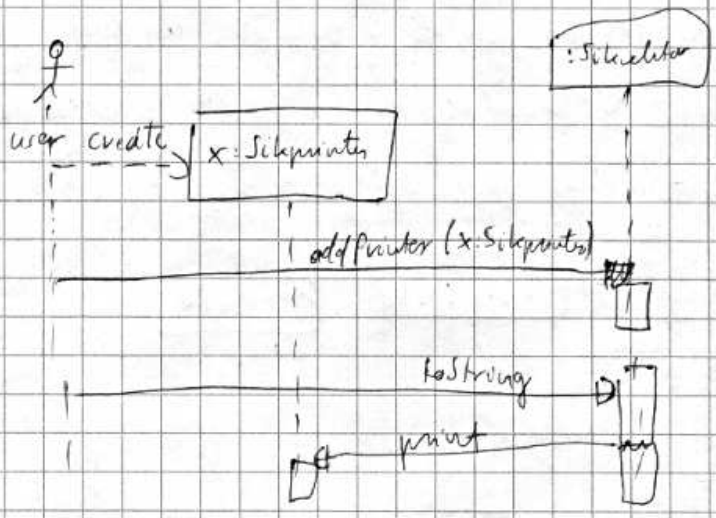


③ Síkbeli vektorok modellezésére legyen egy Síkvektor osztályunk, amelynek egyebek mellett legyen egy toString() metódusa a vektor kiírására. Definiáljon úgy a Síkvektor osztályt, hogy annak újrafordítása nélkül a felhasználó tudjon tetszőleges formátumban kiírni.

⋮

④

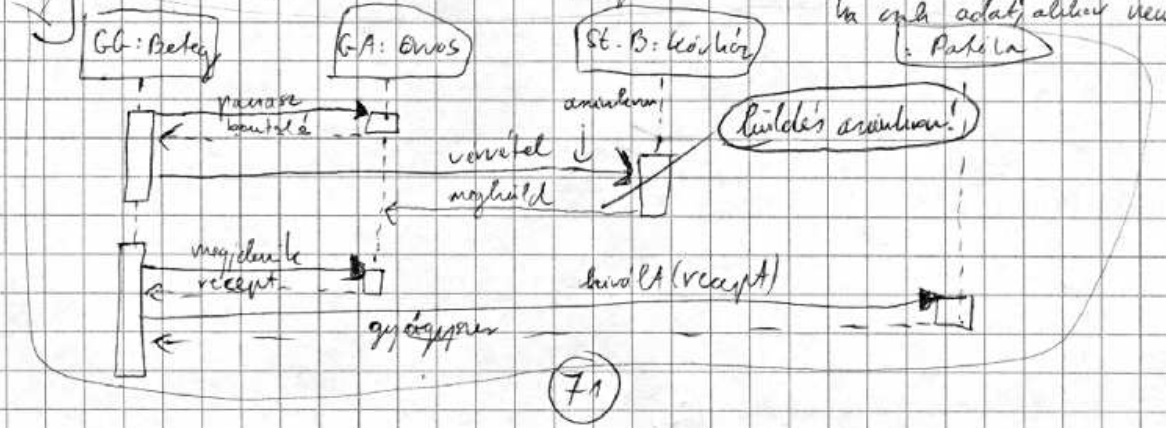




⑤ Géza egyre rosszul érzi magát, ezért elpanaszolja a háziorvosnak, hogy már két hete lázas, és a jobb lábfeje a térdnél duzzadt. Az orvos beutalót ad a kórházba, ahol vért vesznek és a vért vizsgálják. Azeredményt e-mailen küldik a doktornak. Az orvos kapitelőt ír fel és Géza elmegy a patikába kiváltani. UML asztivitás és kommunikációs diagram?

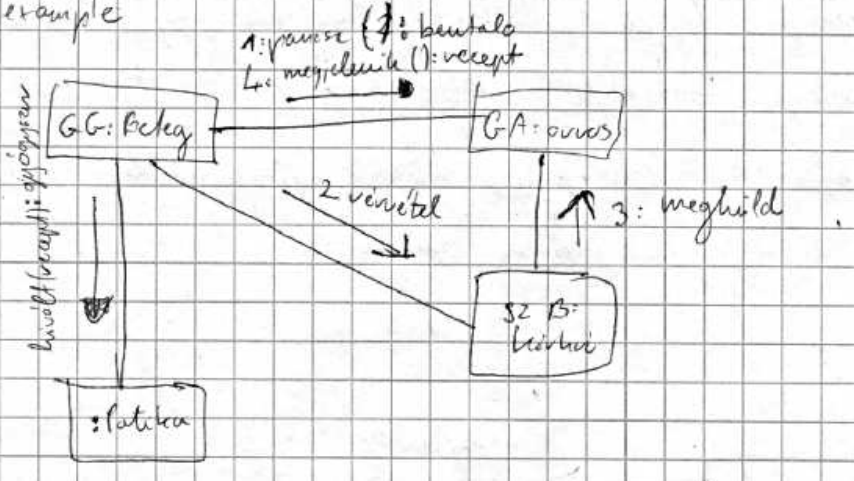
Szereplők: beteg, doktor, kórház, patika ← történet szempontjából van élete, ha csak adatai akarunk nem objektum

⑥



⑦

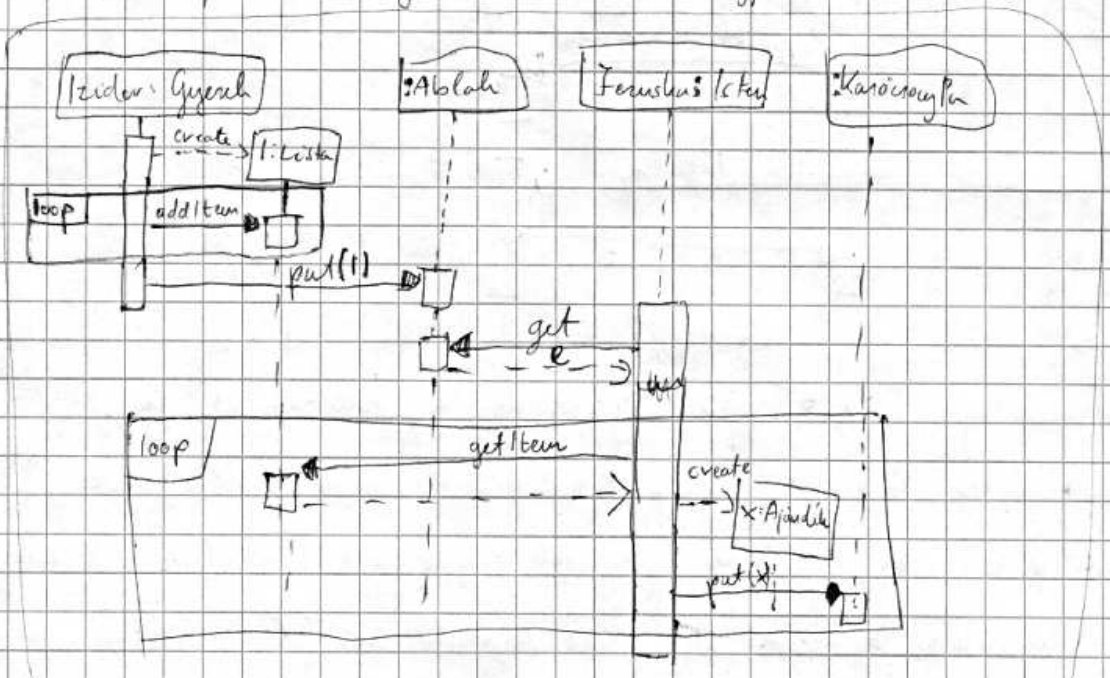
(7) sd example



(11) Kép. UML szekvencia diagram!

Izodor jó fiú volt idén. Decemberben ő is kívánságlistát írt, amit az ablakba tett. Szeretne Jézuska elvenni a listát és elővarázsolja a vajta szereplő kívánságokat és a karácsonyfa alá teszi

(12)



2008.12.03.

(100) Timing Diagram

• vizsgáról

• jan 6, 13, 27

• hétfőn este lesznek fent infók a web lapon

• 14/24 beugró

• papír nem kell

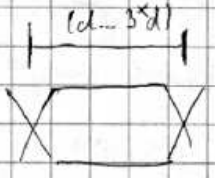
• eredmény e-mailben – pótlelen van kotelezettség

(101) Időzítés

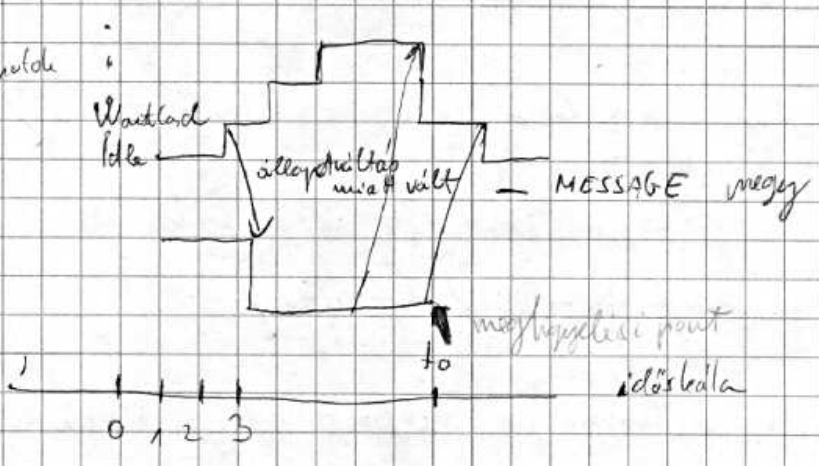
• digites

• vízszintes a lifeline, az idő balról – jobbra halad

• X állapotváltások

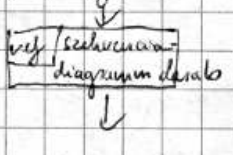


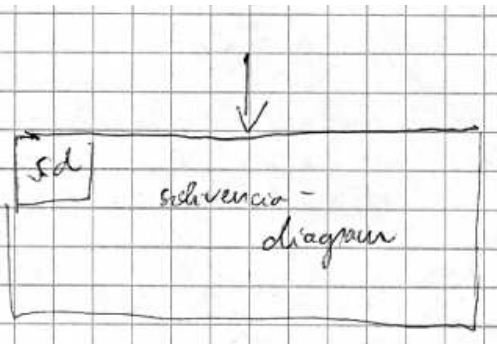
(102) • lúd élettestan (Useré & ACSystemé)



ol: mennyi ideig utazik az üzenet

(104) • folyamatábra jellegű

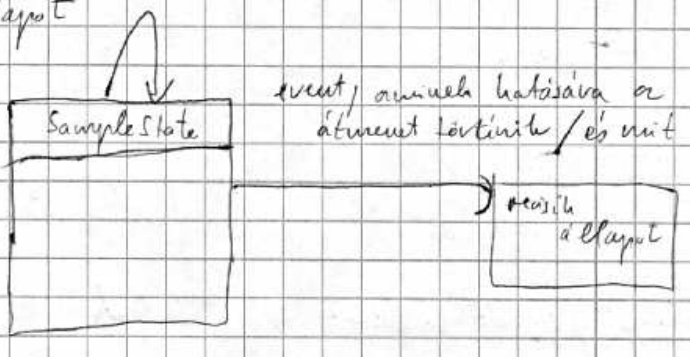




} lehet időbeliségben adni a két lépés között

(107) Állapotgép

- 2 állapot



- entry : entry-action – állapotba lépésnél kell akkor csinálni

  (pl) : boltba belépsz, kiscsönsz – független attól, hogy honnan jössz, hová mész

- exit : exit-action

  (pl) kilépsz a boltból

- ha nyílba mész, akkor exit + valami / entry / on event

  (pl) kutyát hagysz a bolt előtt és ~~vissz~~ ki kell menni pottyugat

  - on event nél nincs exit + entry

- amikor állapotban vagy, az a fölhalt dolog, eseményeket akkor vizsgálsz, amikor két állapot között átmegyünk – 0 idő alatt történik az átmenet elvileg

- gyakorlatban bekövált állapotban is tevékenység kezdődik

  do ...

- ha do-zunk és eventet kapunk, akkor azt kell csinálni és ha végzett akkor nem tudni, hogy folytatódik vagy újra kezdődik !?

• eventuch lehet paraméter ( )

• pl gyorsvonat 6:20-kor indított
paraméter

• event lehet feltételes is { }

(110) Kompozit állapotok

• állapotokat össze lehet vonni

• meglevő állapotokat lefedünk egy nagyobb állapottal

pl: autó sebességének állapotait {R, 1, 2, 3, 4, 5, 6} és lefedjük

{R, 0, előre}

↑
sorrendben 1 vagy 2 vagy .... 6 ezek ez egyik

• új karakteres állapotok létrehozása

(111) 4 ábra ugyan azt mutatja

• initial state - induló állapot

• final state - kilépő állapot

• A tartalma B-t, C-t - ha B-ben vagy C-ben vagyok, akkor A-ban is

• bal fent - p hatására B-ből C-be lépek

• bal lent - ha A-ba megy p, akkor C-be megy az initial state miatt

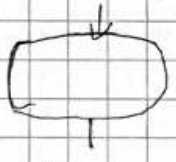
• entry pont ○

• exit pont ⊗

•

állapotok vannak benne

(via ex1)



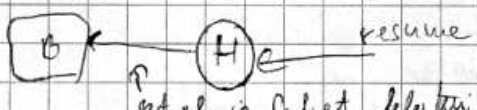
– alternatív ágelés

(via ex1)

(112) Hisztory endikátor

- interrupt miatt ki kell lépni A-ból
  - ha vissza akar térni, akkor nem tudni, hogy A-n belül B-be vagy C-be léptünk

Historia endikátor – ez állapoton belül van!! Emlékszik, hogy hol hagyta abba



ezt el is tudja felejteni ha H-ból működik

- Ha H elfelejt, vagy nem a resume-ra akar valahol lejönni, akkor állítsuk, hogy kivinnent valaki (rosszkor rakott gyökér volt).
  - akkor INVIC állapot jön!!

- (H*) – emlékszik a teljes szerkezetre
- H – nem lát lejjebb, csak a saját szintjén!!

(113) Valódi kollaterális állapotok

- több dimenzióban értelmezett állapotok

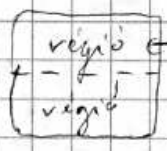
Pl világítás – kint, parkoló-fény, városi világítás, reflektor

- a világítás független a valóstól
- Az autó globális állapota – nincs világít, hátsó ajtó van

↓

független állapotok

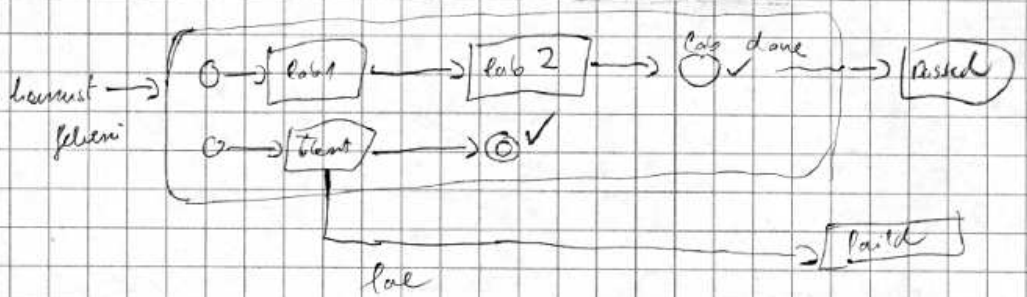
· dimenziók = régiók



<u>Ábra</u>

- tárgyfelvételek
- 2 labort kell elvégezni + 1 tent

labor hat egymás után kell



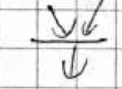
(114) a párhuzamos folyamatok között időnként szinkronizáció kell

- szinkronizáció a régiók között

(116) Aktivitás diagram - FOLYAMATÁBRA

- minden tevékenység egy állapot és akkor lép tovább, ha végzett

(118)

fork  join 

- szétválasztja a folyamatábrát - a részét, hogy melyik tevékenységért ki felelős

(119)

- paraméterként objektumokat is át lehet adni
  - [ ] ∈ objektum állapota

(121) Az UML-ről tíz

- van élet az UML után
- 2 db objektum például UML-ben ellátva

(122) MZ asztali pc sebességjavítás

Part 10

(2) ~~Rational Unified Process~~

~~Work flow~~

(3) • lépések részben rendezett sorozata
- a szoftver felépítését
- iteratív, inkrementális
- usecase vezérelt
- architektúra-centrikus

} VIZSGÁN KELL

(4) • rövid tartalmi leírást készíteni a dél

(5) • használati esetek mögé kiváltséget lehet rakni

(6) •

2008.12.05

(3) UML diagramokat tanít, de a sorrendről nem beszél – erőt próbál mondani valamit a RUP

- usecase által hajtott
- architektúra centrikus
- iteratív, inkrementális

(7) • nagyvonalú áttekintő kép

• Mikor, mit kell csinálni, mi az eredménye és ki csinálja

(8) A RUP három átfogalmazva a szoftverfejlesztő modellt

- alapozó fázis → megvalósítási fázis tanulmány
- kidolgozás ~~előkészítő~~ – analízis, tervezés
- összeállító – megvalósítás
- átvitelre kész használatba

(9) • az egyes fázisokon belül sok iterációs lehet

• az életciklus MUNKAFOLYAMATOKBÓL áll

- fejlesztési munkafolyamatok

  deployment – bevezése

- kiszolgálási munkafolyamatok

  - konfigurációs menedzsment – szoftver mögött álló adminisztráció

- púp magassága ~ munka mennyisége
- environment – ci szükséges eszközök, technológiák beszerzése

(10)

<u>követelmények</u> – use-case modell tartalmi bővítése

<u>analízis</u> – analízis modell – nagyvonalú class-modell / szolgáltatások lényege

+ szekvenciadiagram a use-case-ek való együttműködésre

<u>design</u> – részletes struktúrák, részletes osztálydiagramok / első működési szekvenciadiagramok + package diagramok / komponens vizsgálat

<u>implementáció</u> – részletes kódok, kód

<u>deployment</u> – telepítés, elrendezés

(12) (13)
(11) • melyik modellhez milyen diagram tartozik

(15) (16) (17) (18)
(14) egyes fázisokban milyen tevékenységek kellenek

(16) adatbázis tervek, kiegészítések

(19) követelmény-fázisra tartozó tervmények

---→ függőségre utal

• dom conceptual – nagyvonalú koncepció

• budget – költségvetés

(20)

(22) Use case leírása

a, actor szemlélet

• kik lesznek a felhasználók, kik a szereplők

• funkciók még nincsenek meg

• üzleti folyamatok átszervezése – jó software miatt kevesebb dolgozó kell

b, esemény szemlélet

• milyen események vannak

• új ügyfél, meghal, összeházasodnak ...

• ehhez is lehet átszervezés

3 módon kategorizálható

• mennyire részletes – magas szintű (nem részletes) / kitöltött (részletes)

• logikai / valóságos
↓
technológiától független

hogyan van az együttműködés / megvan adva a kapcsolat technológiája

• mennyire fontos

• alapvető, meghatározó, elengedhetetlen – primary

• vannak olyanok, amik nélkül lehet élni – secondary

• opcionális – menedzsment valamiről (80) lehetőségeket tud számolni

(23) use-case model létrehozása

- ki van benne és ki van kint – a határokat meg kell húzni
- use-case kapcsolatait felderíteni
- kibontani a leghosszúratosabb use-case-eket
- nagyságrendjük

(24) vágsorolásuk

expanded ... – fogalmi ~~tisztított~~ kiterjesztett use-cases

(25) rendes szintű szekvenciadiagram – System seq. diag.

contract – szerződés a szekvenciákban résztvevők leírott

State diag. – állapotgépek: különböző állapotokban ugyanaz az esemény másképp kell reagálni

interakciós d. – objektumok közti működés szekvenciája

- esetek vagy vízióban adatbázissal megoldott a persistencia

(26) Fogalmi model

térképészet elv:

① területen hanyagul nevezhet hanyagjunk

② ki kell hagyni a nem relevans információkat

③ nem létező dolgokat ne adjunk hozzá

(28)

(29) Ami nem eleme a modellnek

- redundáns
- irrelevans
- formátlan, testetlen fogalmakat
- attribútumok

távolodás!

Szükséges ismerni a másikat, mert amiről nem tudunk, azt dolgozni. ?

- a fejlécéről kapott adatokat fel kell tüntetni és nem ASSOCIÁCIÓ, mert

felírod azt kódból!

- analízisben nem nagyon kell görcsölni az asszociációkon

(30) Miből lesz asszociáció példányai:
- valamivé válik, bemegy rá, fogja, használja, szomszédja, ismeri, logol, vágyat, jelent...
- interfészt asszociálhat ki kell dobni

(PE)
- szülő-gyermek kapcsolat
- nagyszülő-unoka - lehet két hív, de lehet rövidzárral is, ha sokszor kell összekapcsolni őket

(31) Mi attribútum, mi osztály?

(33) delegálás: másik objektumnak átadni a feladatot
- az ínas csinálja a munkát, a főnök vállalja el

(38)
1. Ki miért felel
2. az erősség-gélt kell nézni

(40) állapotgép minden olyan keretrendszer elemhez kapcsolódhat ami függ az állapotoktól

(41) Interfészek - miért kell?
- tudásuk valamiről
- képességek valamik

portfóliónálva több dolog lehet#

(42) Architekturális tervezési minták

Facade, Observer, Command, Proxy

(48)

2008.12.10.

Part 11

- 450 házi
  - 90 nem adta be
  - 22-ből nem lett meg

Verifikáció, validáció

(2) ~~Verifikáció: tesztelés~~ ~~validáció: ellenőrzés~~

(3) Verifikáció – jól csináltuk a dolgot

Validálás – jó dolgot csináltunk

- avégett megnézni valamit, hogy jó-e, az már régi kifőtt vau
- egész életciklusban végigvonul a verifikáció & validálás
- jó lenne minél korarabb rájönni, ha nem ezt csináltuk, nem úgy csináltuk ahogy kellett volna
- specifikáció szintjén jó lenne kiszűrni a hibákat

(4) két módnes

(1) Szoftver ~~review~~ tesztelés

- a program futtatása hibakeresés ~~céllal~~ miatt
- a nem funkcionális tulajdonságok csak így derülnek ki
- ez csak akkor jó, ha van kód, amit lehet futtatni

(2) Szoftver review

- statikus ellenőrzés, elemzés, amikor nincs mit futtatni
- több szem többet lát alapon kell átnézni
- a nem funkcionális követelményeket (gyorsaság) review-val nem vizsgálható

ad (2) – definíciók

- review
  - valamilyen anyagot megfognak és megnéznek
- review item
  - az anyag, amit vizsgálnak – kód, specifikáció, UML-diagram
- audit
  - egy review
  - általában külsősöket szoktak csinálni – auditor
    - checklistet pipálgatnak – 2 csillagos szállodából 3 csillagos
  - magas minőségű átnézés
- walkthrough
  - átnézzük
  - valaki kiáll (előre kijelölt) és felügyelő bizottság előtt prezentálni kell / pénzügyi, főnöki, munkatársi
  - kérdéseket lehet feltenni az item-ekről
- inspection
  - walkthrough formalizált formája, kicsi audit
  - az itemeket bizottságoknak osztják
  - átvizsgálása checklist alapján az item-nek
- review meeting
  - meeting előtt 2 nappal kell küldeni az elemeket – pl valami dokumentációs rendszeren keresztül
  - meg kell határozni a résztvevőket
  - szereplők: – koordinátor: – ülésvezető – helyi főnök
    - lead reader: – főolvasó – ő a szereplő
    - scribe: jegyzőkönyvvezető – jó jegyzőkönyvvezető kicsit ér

- felelősségek képviselői
- meg kell hirdetni

(8) • meeting alatt

- kb 2 óráig fest - nem 8 órás kivégzése a lead readersnek
- a probléma feltérképezése a cél
- a vezető megfelelő kérdésekkel között tartja a meetinget
- lead reader szerepel - válcnak
- ki kell derülnie (jegyzőkönyvből is) a felelősnek
  - ki kell jelölni, hogy mit várunk el a felelőstől
  - meg kell mondani mennyire nagy a hiba
  - minor action - csak szólni kell neki, hogy javítsa ki
  - major action - be kell számolni a javításról

(9) • meeting után

- jegyzőkönyv elkészül és szétküldi (ügyített, hogy ki kap)
  - szerepelnek a felelősök

(10) • Review report = jegyzőkönyv

- idő, dátum, ittenlét szereplők
- összejövetel tárgya
- felelős - akció - határidő

(11) Tanács szabályok

- fel kell készülni a meeting-re
- időben normálisan, kooperatívan viselkedni - nem kell védekezni
- nem szabad elkésni !! - nem illik előbb elmenni
- mobil kikapcs
- el kell fogadni a többiek autoritását

(12) Tesztelés

- élő vizsgálat – kódot kézzel végrehajtani
- információkat (specifikáció szerinti működés) ellenőrizni
- ⊕ teszttel azt lehet bebizonyítani, amit akarsz ...
  - ↳ lehet addig tesztelni, amíg célt érünk
- a van hibát azt könnyebb mutatni, mint a nincs hibára

Terminológia

- hiba (error): emberi tevékenységgel kapcsolatos, tevékenység hibája
  pl: elfelejtkezünk valamiről
- fault (bug): emberi hiba (error) következménye a programban
- failure: a jelenség, a tünet – elmúlik a program
  ahogy a felszínen megjelenik a fault

(13) Célok amiért tesztelünk

- együttműködés
- implementálás jó-e
- hibák vannak e?

(14)

- kisebb hiba – nehéz lefuttatni
- amíg a materiális dolgokat lehet „vállalkozásszerűen" tesztelni, addig a szoftvert nem – nem tudni, amerre indulsz találsz-e hibát
- nem vonnak időt a szoftver -ről
- vannak alvó hibák
- ha tudnánk előre hány hiba van a szoftverben az jó lenne...
  - meg lehet becsülni – vannak rá módszerek
    - úgy mint, ahogy a halakat számolják a halastóban

- hibapont 100 halat, megfestik a farkukat és visszaengedik őket
- 2 hét múlva 100-t újra, ha 20-nak festett a farka 20% hal van

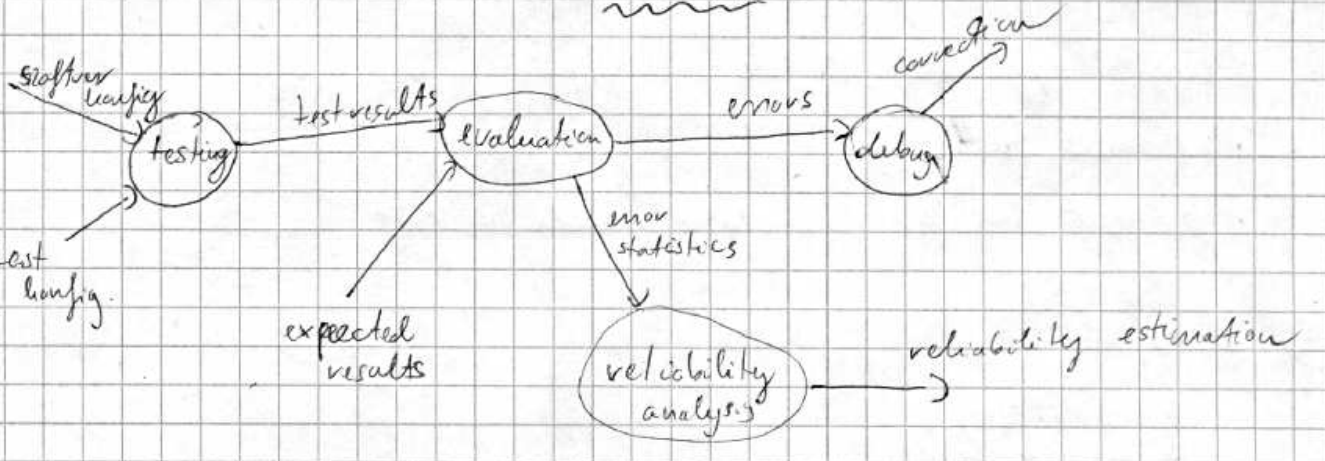
Szoftvernél:

- a minőségbiztosítási csoportban bizonyos emberek dévelt hibákat valamik bele – megnézik, hogy adott idő alatt mennyit javítottak ki a megfestett "hibákból"...

(15)
- 30-50 % a a költségeknek tesztelésre megy el
- bonyolult jól tesztelni
- a tesztelésnek nincs egységes metodológiája
- statikus technikákkal a tesztelést együtt kell használni

(16) Minél korábban eljutunk a kódhoz, annál hamarabb el lehet kezdeni tesztelni

(17) EZ FONTOS: VIZSGÁN



- nem akkor kell kitalálni, hogy mik a jó tesztesetek, amikor már megjelennek a jó, vagy rossz eredmények a futás után

(18) V-model

(19) Unit-test

- fogunk egy egységet és azt tesztjük
- programozó csinálja, aki csinálja

(21) Integrációs teszt

- független csapat végzi – valaki írta a szoftvert
- Top-down integráció
  - felülről – lefelé építhetnek
  - hurkot megfejt → majd később a tartalom
- Bottom-up – alulról felfelé építhetés
  - kell tesztdriver, amiből ~~összeállítható~~ indíthatóó

(22) vendnor-teszt – 76-os olimpiát lejátszották LZ-ből a Moszkvai 80-ra

- stabilan, együtt a teljes vendnor tesztelése a követelményekkel szemben
- teljesítmény & funkcionalitás

(23) elfogadási teszt – átadásnál

formális

- precíz fogalmakkal az átvevővel, hogy mit mutassunk meg
- szerepek, egyeztetett tesztek
- informális (α) teszt
- β-teszt – félig használatlan vihetik a user

(25) Dimenziók

1. funkcionalitás
2. használhatóság
3. ~~megvalósítás~~ megbízhatóság
4. teljesítmény
5. támogathatóság

(26) ad (1)
- funkciók
- biztonsággal kapcsolatos követelések
- ~~vagy~~ a sok adattal végzett teszt

(27) ad (2)
- emberi tényezők – ergonómia
- konzisztens-e a felhasználói interfész – word – bubbla X – excel 1-be
  – excel – bubbla x – összesbe
  (1)

csak egyik lehet

• help és szoftver együttműködése

(28) ad (3)

• mennyire hibatűrő/biztos

• stressz teszt

• struktúra teszt

(29) ad (4) • benchmark

• alaptesztek, amikkel bizonyos részeket tesztelni

ad (5)

• mikor befejezett a teszt?

• végigmentünk 4 ágon

• összes követelmény lefedésével

(32) defekt: hibás működés a change request egy formája (változás-kérelem)

a hiba olyan valamilyen, ami végigvonul M$ követelés hibájában

2008.12.12.

(30) Konfigurálás, installálás

• hány lépésben lehet feltelepíteni a programot – Felhasználó-barátság

(31) ~~Hogyan lehet~~ Áruzzítás Tesztelés mértéke / lefedettség

• követelményekből visszavezetés esetén

• programkódon lefedni – összes ágon végig menni

(32) defect: change request története

• különbség defect paraméterek

(33) Hibák elemzése

- hol az adott gyakorlat során hibázni
- milyen hosszú idejig élnek a hibák
- elfelejt trend – csökkenő e a hibák száma

(34) Teljesítményvizsgálat

- tesztel kapcsolatos vezetői elhajlásokat végzi

(3.6) Fest eljárás

- tesztesetek együttese – előfeltételek, teljességük

(37) Tesztesetek

- elemi funkciókat tesztelnek (action, use-case)

(38) Tesztesetek előállítása

- funkcionalitás alapján
- white box teszt – ismerjük a doboz belsejét, a kódot ismerjük
  - ha kódot javítunk, akkor újra kellene tesztelni 0-ról ← nagyságrendileg nem így van
- black box teszt
  - ekvivalencia particionálás
    - osztályok: objektumból egy osztály
      diszkriminátor 0
      hiperster gyökök
    - 2. fokú egyenletnél
  - speciális értékek tesztje
  - határértékek vizsgálata

(41) JUnit

- jó tesztkörnyezet java progik tesztjére

(3)

Def: vagy rendszer fejlesztésének követése, felügyeletével kapcsolatos tudomány. Milyen elemekből áll a SW, és ezek kezelése

(4) Változás-követés – Windows hibajelentés – (CRM)

- de új funkció is lehetséges vagy új attribútum.

Measurement:

- mérés, logolás, kiértékelések → érdekes tudáshoz lehet jutni

(5) ~~Változás figyelése~~, konfigurációs elemek, azonosítás

- konfigurációs elemek meghatározása – nem követhetünk mindent
- a kommunikáció nem mögöttünk, az is dokumentálva van
  - lehet ellenőrizni, ki mit vett le, ami ide kellett volna tölteni, azt letöltötte-e – nem lehet ötvenkilenc ilyen

(7) Menedzsment

- menedzselni kell a fentebb kijelölt item-eket
- ki vagy jóvá egy változtatást, ki implementálja, ki ellenőrzi az implementációt, ki ellenőrzi, vagy a módosítások a kiadható szoftverben módosultak e
- mik a komponensek, mit változott a komponensek előzőhöz képest – 3 hónappal ezelőtti állapot újragenerálása

(8) Elszámolások

- mit érdemes mérni a projektben
- státuszellenőrzés (folyamatos), audit (mérföldkő) előtt

(10) Elemek távolítása

- el kell dönteni, kell-e verziókövetés
- hogyan azonosítjuk, ha kell
- hogyan távolítsuk az adatot

(11) Mi legyen a 3. fél által (külső, kívülről jövő) jövő anyagokkal mi legyen?

- segédanyagokat, OP-ket, compilerelet kell távolítsuk / a source compiler nélkül nem ér semmit

(12) Minden változik:

- követelmények
- környezet
- software

" Nincs olyan rossz felületelemzés, ami ne változna valóra!"

(13) változás kezelés akkor működik jól ha beleintegrálódik a mindennapi programfejlesztés menetébe

(15) UML activity diagram

(16) állapotgép

(17) Milyen fájlokat, könyvtárakat kezünk fel a felépítéshez.
change management eszköz van a fájlnévben - _végleges, _végleges3...

(18) Baseline: egy adott állapotra leírható a további állapotok egy történő változtatásokkal,
a baseline az alapállás és initial commitnak nevezik.
Távolítás valószínűsége nő!!
- mikor kell új baseline-t csinálni

(20) Kibocsátás menedzsment

- verzió, variáns, release, revízió

(21) • oldalágaél +2 számjegy jön és párathant nem csatlak

(22) CVS
Subversion } verzió kontroller

• kommunikációra webes rendszert