Mi az a szoftver?

- Hardvervezérlő utasítások sorozata
- Programok és a hozzátartozó dokumentáció, még a programok fejlesztéséhez, fenntartásához és működtetéséhez szükséges

⇓

- nem fizikai dolog
- modellez
- mérnöki munkát igényel
- folyamatosan bővülő komplexitás
- ötlet

Szoftverfejlesztés problémái:

- kereslet magasabb mint a kínálat
- emelkedő minőségi elvárások → Helyesség* Hordozhatóság, Megbízhatóság, Teljesítmény
- visszamaradt (legacy) szoftverek megkerülhetetlensége,
- régi szoftver újrafelhasználásának szüksége

* Helyesség: Azt csinálja amit kell, ahogy le van írva a dokumentációban

Megbízhatóság: Nem omlik össze különleges esetekben sem.

Teljesítmény: sebesség, hardverigény, stb.

Szoftverfejlesztés céljai:

- programírás hatékonyságának növelése
- szoftver minőség javítása
- olcsóbb fejlesztési és fenntartási költségek elérése
- fejleszthető szoftverek előállítása
- pontosabb költség- és hozzáfelmérhetőség támogatása
- Szoftverfejlesztési folyamat automatizálása

mérnöki munka:
- költséghatékony megoldások előállítása
- gyakorlati problémákra
- tudományos ismeretek alkalmazásával
- építve oldjanak
- az emberiség szolgálatában.

Szellemi + Termelés = Kereskedelmi ~~Fogyasztó~~ termelés
Kereskedelmi termelés + Tudomány = Mérnöki tervezés

Szoftverfejlesztés kudarca..
- lemaradás ütemtervtől
- ~~nagy~~ becsült árlépő költségek
- megbízhatatlanság
- fenntarthatósági problémák
- gyenge teljesítmény

Okok: - jobb sw vagy hw kell(?)
- túlzott sw-függőség → magasabb elvárások
- gyenge minőségű szoftver előállítás
- régi szoftverek támogathatatlansága

Szoftvertervezés.. menedzsment munka.
tudományos
technikai
jogi
gazdasági
technológiai

People	Emberek
Problem	Probléma
Process	Folyamat
Product	Termék

II. Szoftverfejlesztési folyamat

Minőség: Egy termék ~~tulajdonságainak és képességeinek~~ jellemzőinek és tulajdonságainak összessége amely meghatározza a program azon képességét hogy megadott igényeket kielégítsen.

Ezek lehetnek: Megfoghatatan minőségjellemzők, vagy:
- felhasználói minőség
- előállítási -"-
- termék -"-
- árérték arány

Folyamat: nyersanyagok (input) felhasználóknak hasznos termékké (output) alakítása
emberek által módszerek, eszközök, és felszerelés felhasználásával.
ismeretek

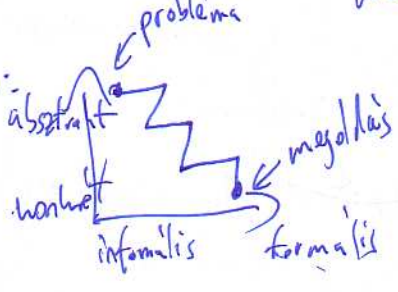
Folyamatmenedzsment elve:
Egy termék minősége arányos az őt előállító folyamat minőségével.

Fejlesztés, javítás:
- jobb minőség
- nagyobb hatékonyság
- olcsóbb
- rugalmasabb
- ~~fejlesztők~~ előállítói elégedettség növelése

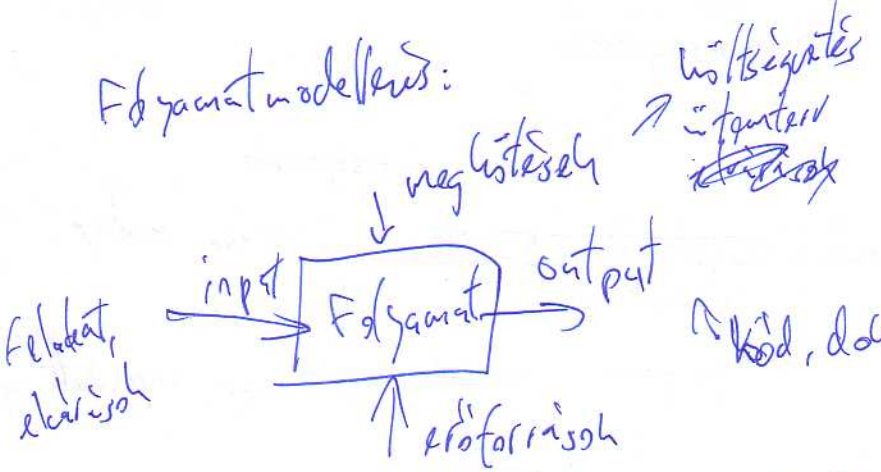
Absztrakt vs. konkrét: Mennyire magával a problémával és illetve annak egy elvonatkoztatott formájával foglalkozunk

Formális vs. informális: Mennyire kötött, egyértelmű módon fogalmazzuk meg a megoldási lehetőséget.

Elméletben:



Folyamatmodellezés:



→ Folyamat részfolyamatokra bontása

— x —

Termék életciklusa:

I. Igénymeghatározás és felmérés (Requirement)
- mi a probléma?
- milyen technológiai/társadalmi/gazdasági kötöttségek

II. Specifikáció (Specification)
- mi a megoldás?
- milyen rendszer képes ezt megvalósítani?

III. Tervezés (Design)
- hogyan épül fel a megoldás?
- mik a részfeladatok?
- miket kell elvégezni?

Megvalósítás

IV. ~~Kódolás~~ és építés (Implementation & Construction)
- A megtervezett lépések végrehajtása

V. Ellenőrzés (Validation)
- tudja a felhasználó használni a megoldást?
- valóban megoldás a problémára?

VI. Fenntartás, karbantartás (~~Maintan~~ Maintenance)
- Milyen fejlesztések, változtatások szükségesek?

4

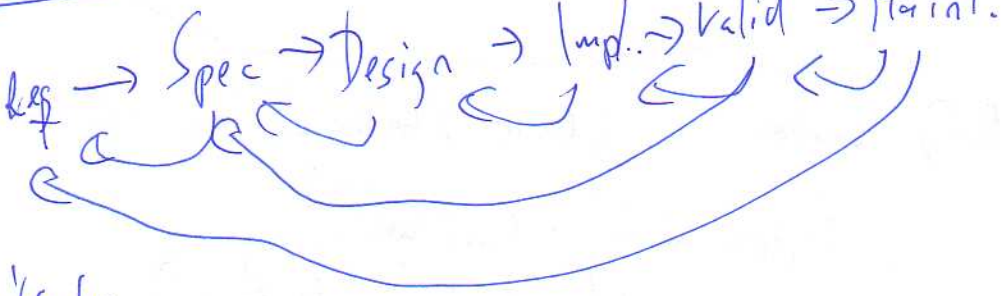
Fázisos modellek

Lineáris modell:

Req → Spec → Design → Impl. → Valid. → Maint.

folyamatosan növekvő hibajavítási költségek,
kisebbek maradnak tervezési hibák is

Vízesés modell:



Átadható ~~Elvek~~ Teljesítések:

~~Req~~ Req: Rendszerdefiníció, ~~Kar~~ Megvalósíthatósági jelentés

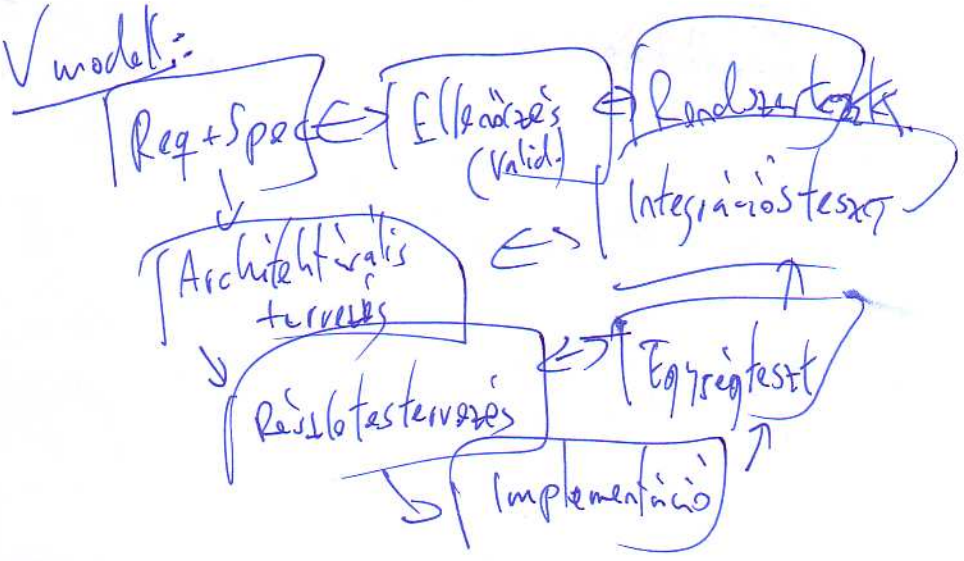
Spec: Megoldásleírás, elvárásdefiníció, előzetes felhasználói dokumentáció

Design: Architektúra terv, ~~dokumentáció~~ felhasználói dokumentáció, ellenőrzési terv

~~Vati~~ Imp.: ~~Kód dokumentáció~~ fejlesztői dokumentáció, elfogadhatósági vizsgálati terv

Validation: Teszteredmények, ~~Fejl~~ fejlesztés terve, projektértékelés

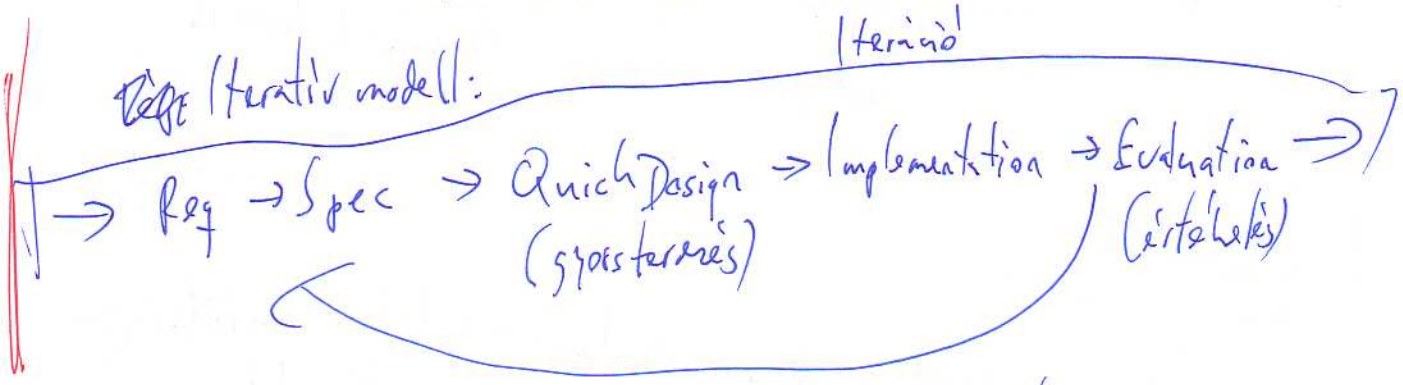
V modell:



fázisos életciklusmodellek problémái:

- nem tükrözik a valós projektet (mestat csak fokozatos)
- nincs meg egyszerre az összes elvárás
- türelmetlen felhasználó (csak a legvégén kap futó kódot)

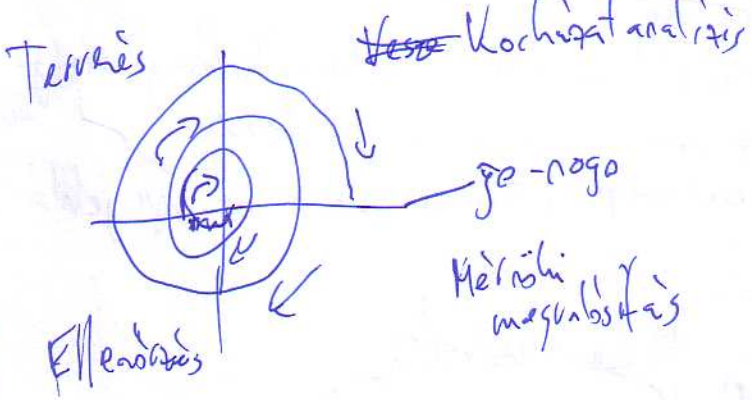
Régi iterativ modell:



RAD modell: (Gyors Alkalmazás fejlesztés)

Üzleti modell (mi kell?)
Data modell (miről oldható meg?)
Process modell (hogyan csinálj/uk?)
Application ~~generation~~ (előállítás)
Testing (ellenőrzés)

Spirális modell:



Capability Maturity Model

CMM

Minőségi mérnöki tervezési folyamat mérésére alkalmas szervezési fejlesztéseket tartalmaz

- ~~menedz~~ felsővezetésnek célkitűzéseket biztosít
- kijelöli a folyamat fejlesztés lehetséges szempontjait
- megbecsüli a jelen és jövőbeli előállítási képességeket
- ~~iparszerte~~ iparszintű összehasonlítást biztosít...

5 szint.

1. Kezdeti (initial) ↓ + betartott folyamat
2. Megismételhető (repeatable) ↓ + stabil, konzisztens folyamat
3. Meghatározott (defined) ↓ + megjósolható folyamat
4. Szabályozott (managed) ↓ + javított folyamat (folyamatosan)
5. Optimalizáló (optimizing)

Kezdeti: ötletszerű fejlesztés, gyakran kaotikus eredmény egyéni teljesítménytől függ

Megismételhető: léteznek menedzsment feladatok, követhető idő és költség le vannak fektetve a gyakorlatok alapjai, hasonló feladatnál ugyanolyan eredménnyel elvégezhetőek.

Meghatározott: a fejlesztés lépései szabályozottak és dokumentáltak a projektek ugyanahhoz a standardekhez igazodnak a fejlesztés lépései részfolyamatokra vannak bontva, ezek struktúrája ismert és kontrollált.

Szükséges: A folyamat és a termék folyamatos minőségellenőrzése megoldott

Mennyiségi

A folyamat minden részegysége meghatározott, mérhető és ellenőrizhető

A problémák a létrejövésük helyén észlelhetők, a haladás pontosan ellenőrizhető.

Optimalizáló: A részletes és folyamatos ~~minőségellenőrzés~~ minőségellenőrzés eredményei lehetővé teszik a ~~tervezett~~ folyamat fejlesztését, újításokat és megvalósításokat ellenőrizve építhetők be, a javítást elősegítő lehetőségek folyamatos kipróbálás alatt állnak.

CMM

1. Kezdeti
- nincs menedzselési procedúra
- nincs projektterv
- nem ellenőrzött
- ötletszerű fejlesztés → eredmény (minőség nem garantált)

2. Megismételhető
- a projekt sikeresen megismételhető
- informális (nincs formális fejl. modell)
- követelményekre menedzselve (adatm., felülvizsg., változtat)
- menedzselt és felügyelt projektterv

3. Definiált
- ~~formális~~, definiált és dokumentált vállalati fejl. folyamat
  - ↳ javítás lehetősége
- formális procedúrák az ellenőrzésre
- fókusz a fejl. folyamaton
- fejlesztés lépései vállalatokra bontva

4. Menedzselt
- formális program
- kvantitatív adatgyűjtés
- mérési eredmények beépítése a folyamatjavításba (minőség ell.)
  - ↳ minden vállalat mérhető, ellenőrizhető

5. Optimáló
- vállalat kötelezettséget vállal a fejlesztési folyamat állandó javítására
- folyamatjavítás integrált része a vállalat tevékenységének
  - ↳ folyamatos ellenőrzés és javítás

III. Projektmenedzsment

Csoportszervezés:
Alapelvek:
- hevesebb, jobb emberek használata
- személyhez szabott feladatok kiosztása
- hosszú távon megéri kihozni a legtöbbet
- megéri harmonikus, jól együttműködő csapatot összeállítani
- aki nem való a csoportba, azt ki kell dobni.

Szoftverprojektterv célja:
egy keret meghatározása, melyen belül a manager mozoghat, megbecsülve az erőforrás - idő - és pénzigényt.

Tervezés:
Tervezők és fejlesztők ⇄ manager
(helyzetjelentés / kiosztás erőforrásallokáció)

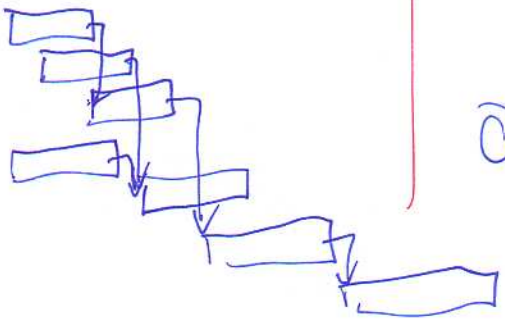
erőforrások: Ember
Hardverelemek
Szoftver segédeszközök

szabályozók: időigény, információ (pl. dokumentáció), minőség, pénz

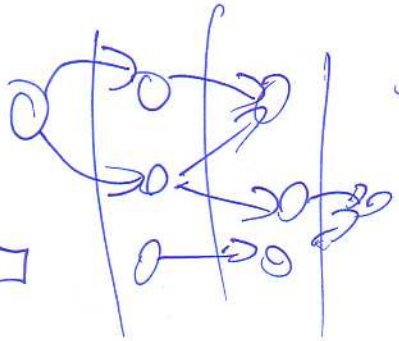
- folyamat részfeladatokra bontása függőségek meghatározásával, párhuzamosítás
- erőforrásallokáció

→ PERT vagy Gantt diagram (adott!!)

Gantt:



Pert:



stb.

Kockázatanalízis

Kockázat: Váratlan ~~esemény~~ ~~egy fok valamilyen valószínűséggel~~

projekt kockázat → idő- vagy erőforrás beosztást befolyásolják

~~egyéb események valamilyen valószínűséggel~~

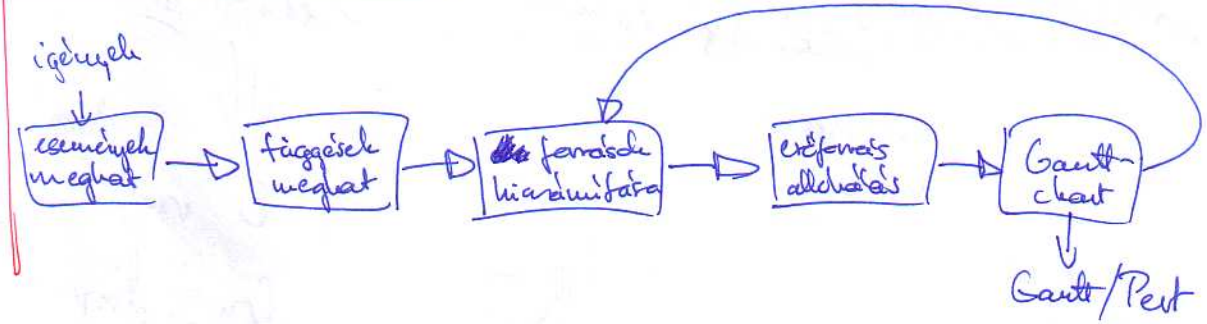
termék kockázat → a szoftver minőségét vagy teljesítményét befolyásolják

üzleti kockázat → a szoftverfejlesztő cégre ható piaci jelenségek

* Kockázat: Annak a valószínűsége hogy valami váratlan kimenet ~~vagy~~ negatívan hat a projektre

Kockázatmenedzsment:
- Azonosítás (mik)
- Analízis (mekkora esély, milyen súlyos)
- Tervezés (hatáselhárítás és minimalizálás)
- Megfigyelés (fellépő kockázatok megfigyelése)

* Időbeosztás készítése



IV. Probléma meghatározás (Requirement)

Minél később találnak meg hibát, annál drágább kijavítani
↳ hibák jelentős része problémameghatározás

- Elvárás definíció: rendszerleírás a felhasználóknak
- Elvárás specifikáció: ~~Szerződés~~ megállapodási alap a vásárló és a készítő közt

↳ Viselkedési elvárás: mit csinál
↳ Nemviselkedési –"–: milyen tulajdonságokkal csinál valamit

- Elvárások: természetes nyelven fölösleg nélkül, strukturált formában, ellenőrizhető állítások formában.
  - helyes (minden elvárás kell)
  - egyértelmű
  - teljes (minden ~~ott~~ ami kell ott van)
  - ellenőrizhető
  - érthető
  - módosítható
  - nyomonkövethető (a megoldás menetében ~~látsz~~ láthatóvá teheti az elvárás elemeit)

Irányelvek
- Principles:
  - információtartalom: adat és vezérlő objektumok amik a lehetséges inputokat adják a rendszernek
  - információáramlás: adat és vezérlés változása a rendszerben
  - információ szerkezet: • mit kell a systemnek megcsinálnia?
    • hogyan viselkedik a szoftver?

Információ capture

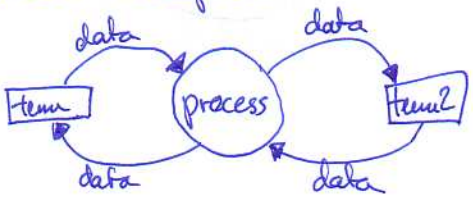
- data — dataflow
- process — function
- control — control flow
- kommunikáció
- konkurencia
- időzítés — szinkronizáció
- kapcsolatok
- valódi
- history

V Specifikáció

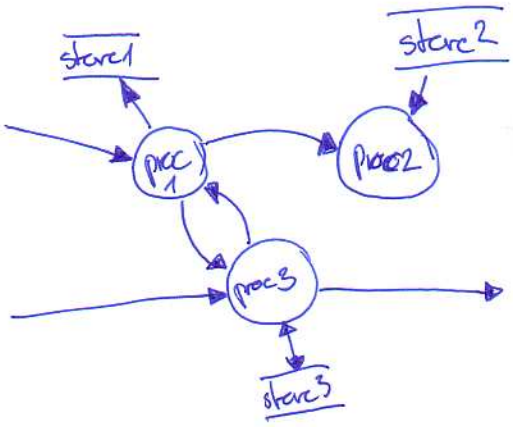
DFD

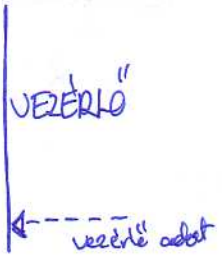
- dataflow
- process
- store
- terminator

⇒ Context-diagram

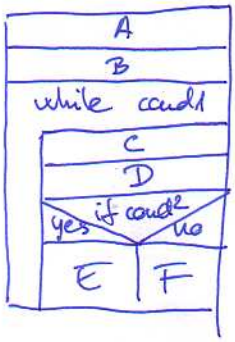


⇒ ~~Functional model~~ Internal process





Nassi-Schneidermann diagram



⇒ pseudokódban:

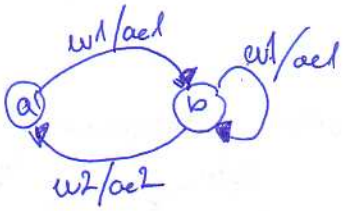
```
begin
  A;
  B;
  while cond1 do
    begin
      C;
      D;
      if cond2 then E
               else F
    end
end
```

Entitás Relációs Diagram

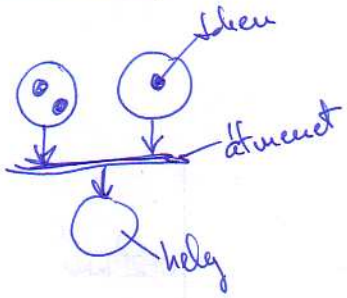
- ent. — entitás / adat
- — } kapcsolat
- ◇ } [redacted]
- ← [redacted]tább is) kardinalitás
- ○ modalitás (van v. nincs)

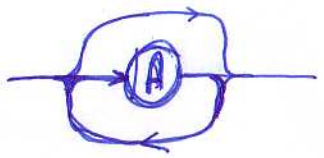
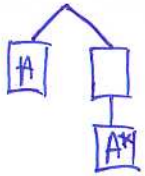
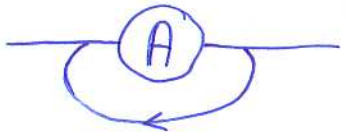
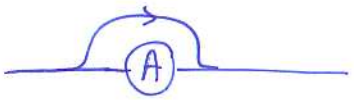
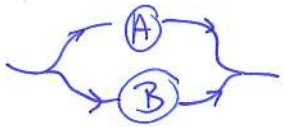
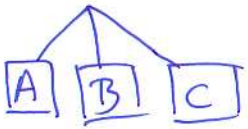
Behavioral modeling

	ev1	ev2
a	b/ac1	
b	b/ac1	a/ac2



Petri-háló



	Alg. def.	DTD	BNF	JSD	syntax gráf
0....n	{A}	A*	{<A>}		
1...n	A+{A}	A+	<A>{<A>}		
0..1	O{A}1	A?	[<A>]		
VAGY	[A|B]	(A|B)	<A|B>		
SEQ	+	,	L1		

VI Tervezés

Tervezés: megálmodni, megtervezni és specifikálni a készülő szoftver belső szerkezetét és a folyamatainak részleteit

Célok:
- belső szerk. és folyamatok specifikálása
- tervezési döntések feljegyzése, választások megindoklása
- véglegesíteni a tesztelés tervét és a user manuált
- ~~implementációnak~~ tervrajzot adni az implementációhoz, teszteléshez és karbantartáshoz

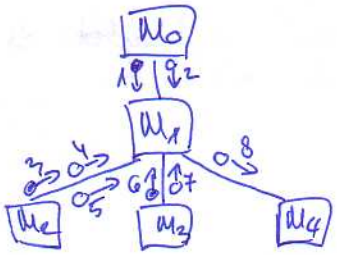
→ Architektúra tervezés:
- rendszer ~~struktúra~~ komponenseit finomítani
- belső folyamatok meghat.
- ~~a~~ magas szintű folyamatok → részfolyamatokká
- belső adatfolyam és feldolgozás
- figyelmek közti kapcsolatok és kapcsolódási pontok meghat.
- adatfolyam
- adattár

→ Részletes tervezés:
- algoritmusok spec. amik figyelmet impl.
- adatstruktúrák amik adattárakat valósítanak meg
- valódi kapcsolat az adat és a figyelők közt

Szempontok
- ∅ csőlátás
- implementálható, analizálható
- a kereket már feltalálták, kösd
- problémával foglalkozzon
- facilitate change
- számítsunk a váratlanra
- nem kell!
- minőségre van
- reviewr

Modularity – Structure diagram

pl:



Coupling (csatolás)

→ ha már szétszedtük a rendszert modulokra, és azok közt relációk létezhetnek más dekompozíció → más relációk

→ a belső kapcsolatok megmutatják egy változás bekövetkezését

⇒ fenntarthatóság fontos indikátora

→ tervezési minőség fontos tulajdonsága

↳ Dimenziói:

- mi kommunikálódik? (adat, stamp, control stb)
- kapcsolat mérete
- kapcsolat ideje

Strukturált Programozás

- Top-down
- inkrementális tervezési stratégia a problémák feldarabolására
  1. tervezési folyamat szisztematizálása
  2. moduláris programtervezés
  3. keretrendszer biztosítása a könnyebb problémamegoldáshoz
- dekompozíció
- lépésenkénti finomítás

pl: JSP

~~VI. [illegible]~~

Absztrakció → csak a lényeggel foglalkozik, a többit elhanyagolja,
egyszerűsíti a problémát → kontrollálja a részletezés szintjét és mennyiségét
~~részletezett részletesen~~
→ megkönnyíti a kommunikációt

Típusai:
- procedurális – nem fejtjük ki hogy egy procedúra mit csinál (csak megnevezzük)
- adat – nem fejtjük ki, hogy egy adatszerkezetnek mi a tényleges szerkezete, hanem elnevezzük
- vezérlés – nem fejtjük ki, hogy egy vezérlési folyamat hogyan működik ⇒ jelöljük a kívánt hatást

Egységbezárás (encapsulation) → összegzi néhány objektumot logikailag vagy egy fizikai fióldba (process: ezt csináljuk, entity: egy objekt amivel ez megtörtént)

Dekompozíció: Egy program teljes komplexitása nagyobb mint a kisebb részegységek és ~~azok~~ összekötésének kompozíciója.

Modularitás → feladatmodulok
- fan in – meghívó modulok száma
- fan out – meghívott modulok száma
- döntés hatókör ~~[illegible]~~ miről tud önálló döntést hozni??
- irányítási ~~[illegible]~~ hatókör (mit tud csinálni)

szedjük szét modulokra a nagy bonyi feladatot hátha a modulokat könnyebb kezelni

=*

Kohézió: egy modul elemeinek összetartozása. Mennyire értelmes adott funkciókat egy modulba zárni.

erős ↑↓ gyenge (FONTOS)

- Funkcionális – uazt csinálják
- Szekvenciális – egymásután csinálják (output → input)
- Kommunikációs – ugyanazon csinálnak mást
- Procedurális – kb. egymásután csinálják (output ↛ input)
- Időbeli – egyszerre kell csinálni
- Logikai – van közük egymáshoz (laza kapcsolat)
- Véletlen – nincs.

~~[illegible]~~ kontrol hatók. < döntési hatók. ↳ döntési hasítás

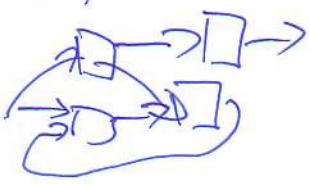
Családok: ~~Hívó~~ Hívott kommunikál két modul egymással:
data (adat)
stamp (összetett adat)
control (modul vezérlés)
common (közös adatterület)
content (közös kód)

Jackson Structured Programming JSP

1. I/O adatok definíciója
2. struktúrák felbontása
3. futtatás: információ előállítása
4. kódtranszformáció

Szoftverarchitektúrák

I. Adatfolyam
- Batch (Output → Input → Output-ig stb.)
- Cső/szűrő (Pipe&Filter) - felosztja részfeladatokra
  ↓ kapcsolat ↓ adattranszformáció majd adott output/ja a következő



p!

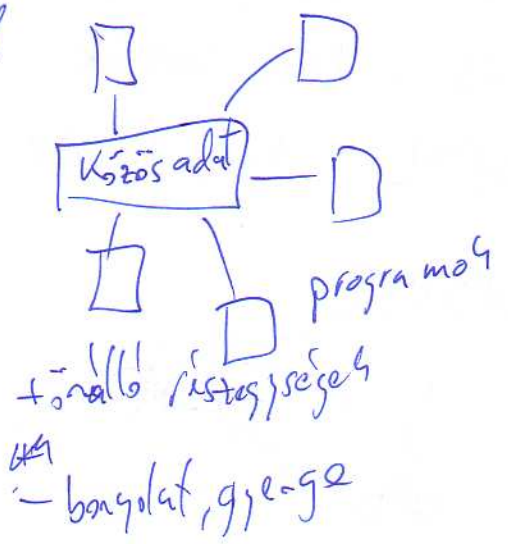
+ újrahasznosítható
+ párhuzamos
- nem interaktív
- túlságosan egymásutáni lépések

II. Hívásos rendszerek
- Szabványos rendszerek
- OO rendszerek → majd később
- Réteges rendszerek

III. ~~Adat~~ Adatalapú rendszerek ↓

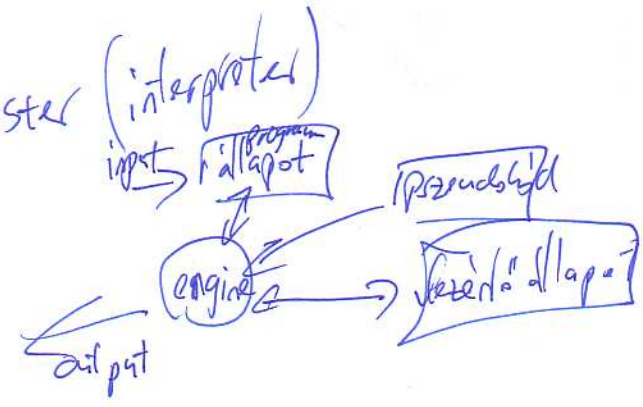
III. Adatalapú rendszerek

- Adatbázisok
- Hypertext (linkelt tartalom)
- Blackboard



IV. Virtuális gépek

- Fordítóprogramok (interpreter)



Rétegzett architektúrák

Egyszerű: alulról indul, felülre készítenek

Vízesés

Függőleges: Azonos feladatokat egymás alá

Kliens-szerver architektúra:

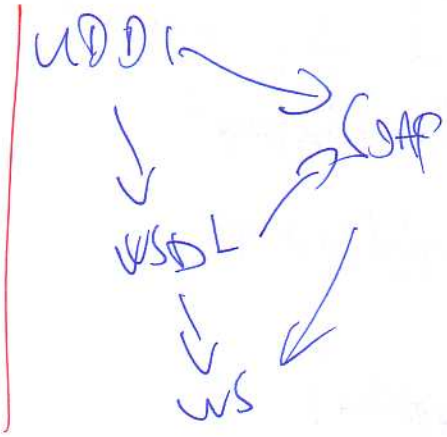
GUI

BOM (BUSINESS OBJECT MODEL)

DB (DATABASE)

SOA - service oriented architecture:

UDDI - Universal Description Discovery and Integration
SOAP - Simple Object Access Protocol
WS - Web Service
WSDL - Web Service Description Language



VII. JSD

Model:

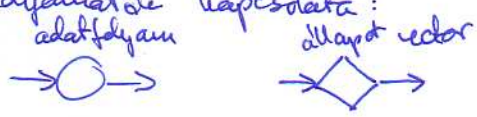
- alkalmazási terület = entitások csoportja
- Entitás: a világ egy létező eleme, ami eseményt generál vagy elszenved
- Modeling = entitások felfedezése
- modellezés mint funkciók lényegesek
- a model stabilabb mint a követelmények
- fenntarthatóság

~~Folyamat:~~

Folyamat:

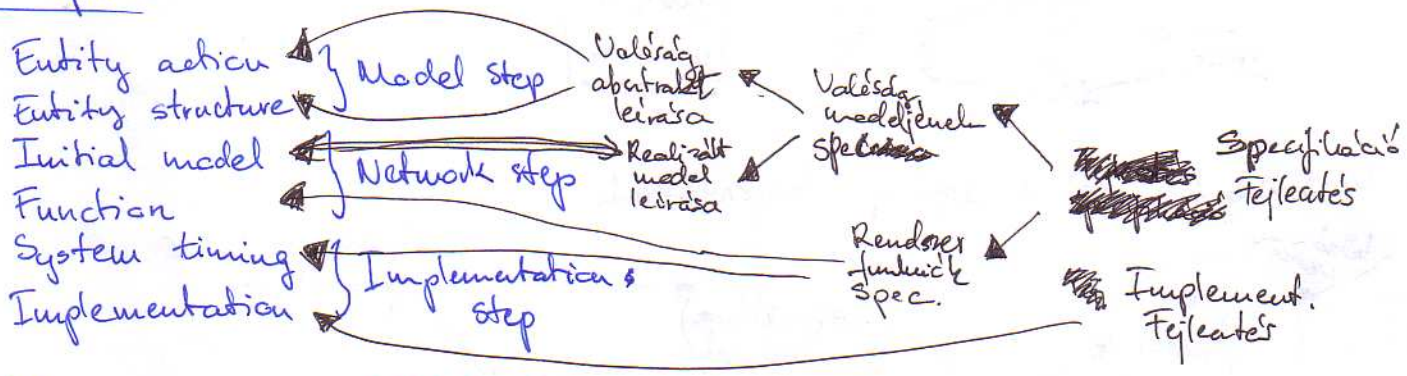
↳ folyamat például:
- program szöveg
- állapot vektor (text pointer)

↳ folyamatok kapcsolata:



↳ folyamat implementáció
- specifikáció követtem implementálása
- tevezés ütemező
- állapot vektor szeparálása
- program szöveg feldarabolása

JSD lépései:



1. Entity action — entitások és eseményeik azonosítása

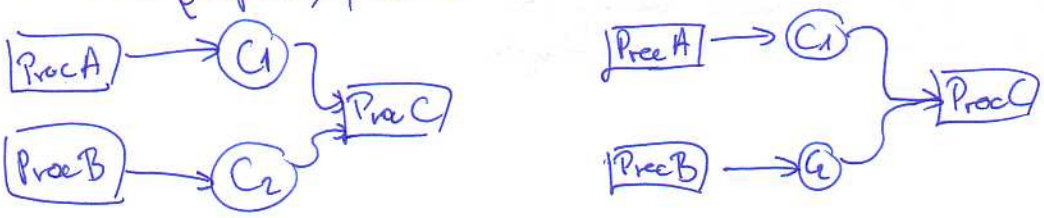
- Entity? → főnevek, dolgok, szereplők, az adott területről stb....
- Action? → igék, elemi dolgok, külső világból

2. Entity structure

- entity life history alapján processz-hálózat felvétele
- ~~[illegible]~~ közös tevékenységek
- Jackson Structure Diagram az entitásokra

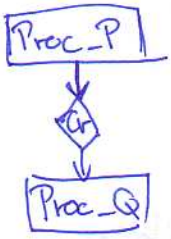
3. Initial model

- System Spec. Diagram (SSD)
- adat-folyam kapcsolatok
  - ↳ végtelen ~~FIFO~~
  - ↳ pl: read (suspend), write

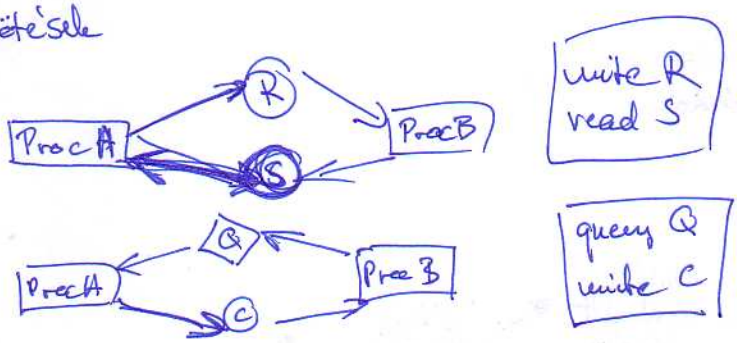


- state vector kapcsolódás
  - ↳ pl = prec_Q indítja a folyamatot

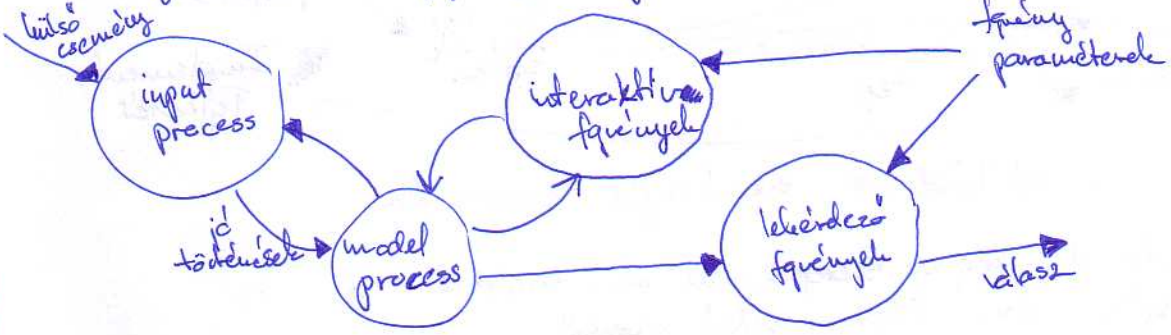
getSV:



- megkötések



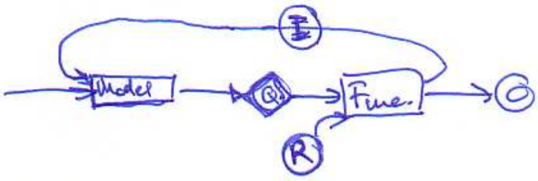
- Valós világ + model + fgvény kapcsolódások.



4. Function — process 2 háló kibővítése funkció-processmodell

- fgvények & modellek :

- iteratív fgvények :



5. System Timing

- kimenetek megkötései
- getSV gyakorisága
- model-process frissítési frekv.
- time-grain marker olvasása
- határidők

6. Implementation

- processek ütemezése
- inverziók

XRUP

Rational Unified Process

- lépcsőzetes & iteratív → több ciklusos analízis, tervezés és építés
  → korai feedback a megrendelőnek és developernek
  → fokozatosan újraalkalmazható csíkok
  → Req. változtatható ⇒ piaci megfelelés
  → korai átfogó integrációs tesztelés
- use-case orientált → Req. megjelennek a use-case-ben
  → becsléseket a use-case befolyásolja
  → időbecslések use-case alapján
  → koncepció, osztályok és tesztek use-case alapján
- Architektúra kp-ú → kevés rendszerarchitektúra kérésére ösztönöz
  → rétegelt arch.
  → elrendezés kézi kicsi kapcsolat
  → könnyen érthető
  → robosztus, sokat bír, nagy rs-re is alkalmazható
  → sok újrafelhasználható komponenses
  → fontos és kockázatos use-case-ek alapján vezérelt
  → jól definiált interfészek

→ Strukturált folyamatok:
- Mikor? → életciklus felosztása fázisokba
- Mi? → jól def. történések
- Mi kénül? → egy bizonyos elemcsop. termékei
- Ki csinálja? → HR

→ Életciklus fázisai:
- Inception (Előkészítés / kitalálás): piaci igény felmérése, projekt behatárolása, projekt elképzelés specifikálása
- Elaboration (Kidolgozás / kifejtés): források és tevékenységek meghatározása jellemzők specifikálása, architektúra tervezés
- Construction (Megvalósítás / megépítés): működő rs megépítése és tesztelése az előzetes tervnek megfelelően ismételt lépésekben
- Transition (Átmeneti / átadás): termékek átadjuk a user-nek (legyártás, kiszállítás, training)

→ Munkafolyamatok (workflow):

Requirements (igénymegfelelés)
- vázlatos terv
- előzetes nyomonkövetési jelentés
- igények meghatározása
- meglogalmazások lejegyzése
- prototípus implementálása
- use-case meghatározása (a fontosakat)
- vázlatos fogalmi model
- vázlatos rendszer architektúra

- Mit kell tudnia a termékek?
- Ezt olyan formában leírni hogy a vevő és a dev. team is könnyen értse
- Tartalma:
  - áttekintés
  - vevők
  - célok
  - rendszer funkciói
  - rendszer tulajdonságai
  - feltételezések
  - kockázatok
  - szójegyzék

Analysis (Analízis)

- dengedhetetlen use-case elk def.
- use-case diagramok finomítása
- fogalmi modell finomítása
- nézőpontok finomítása
- rendszernek szekvencia-diagramokat rajzol
- működési szerződés
- állapot diagram

Design (Tervezés)

- valódi use-case -ek
- jelentések meghat., UI, storyboardok
- rendszer arch. meghat.
- interakció-diagram
- osztálydiagramok
- adatbázis-séma

Implementation (Megvalósítás)

- class & interface imp.
- metódusok imp
- ablakok imp
- jelentések imp
- adatb. séma imp.
- tesztek kód írása

Deployment (Telepítés)

- párhuzamos futás & keresztezések megállapítása
- tech. doku befejezése
- user doku befejezése
- system test
- acceptance test
- document test
- training
- support létrehozása
- telepítés

Use-case -ek

- actor-based: → egy rendszerhez v. szervezethez tartozó actorok azonosítása
  → minden actorhoz azonosítani kell melyik folyamatot indítja, vagy módosítja
- event-based: → külső események azonosítása amire a rendszernek válaszolni kell
  → összekapcsolni a megf. actorokat, eseményeket és use-case-eket

Típusai:

- magas szintű v. bővített — high level, extended
- alapvető (tech. független) v. valódi — essential, real
- elsődleges v. másodlagos v. opcionális — primary, secondary, optional

Use-case készítése

1. rendszer funkciói alapján ⇒ rendszerhatárok majd actorok és use-case-ek megadása
2. magasszintű use-case-be írd mindet és kategorizáld
3. use-case diagram
4. use-case összekapcsolódása
5. kritikus, befolyásos, és kockázatos use-case-ek kibővítése
6. ideálisan, a valódi use-case-ek csak a Tervezés folyamatban jönnek elő
7. sorrendezés ⇒ :
   1. az architektúra nagyjára hatnak
   2. kockázatos, időkritikus, komplex
   3. kutatást igényel, vagy új, kockázatos ~~[illegible]~~ fejleszést
   4. elsődleges üzleti vonulatot képvisel
   5. közvetlen support, növekedett bevétel, csökkent költségek

Ami ismételhető : analízis —bővítés→ ~~[illegible]~~ tervezés (← valódi)

Fogalmi model

~~[illegible]~~

~~1. Lehetséges koncepciók listája (kategória vagy~~
~~kategóriák:~~
- ~~kézzelfogható objektumok~~
- ~~specifikációk, tervek, leírások~~
- ~~helyek, tranzakciók~~
- ~~emberek szerepe, szervezetek~~

1. Lehetséges koncepciók listája
   - ~~kategorizálás~~
   - ~~[illegible] a feleslegest~~
2. Maradj a jó termnél
3. Fogalmi model rajzolása
4. Koncepciók közti asszociációk megrajzolása
5. Fölösleges vagy helytelen assz. kidobálni
6. Tulajdonságok megadása.
7. Generalizáció - specializáció hierarchiák
8. szójegyzék

Gyakori asszociációk:

- A fizikai/logikai része B-nek — part of
- A fizikailag/logikailag B-ben van — contained
- ~~A~~ ... member of
- owned by
- next to
- uses / manages / communicates
- related to a transaction
- known / logged / reported / recorded / captured

System Sequence Diagram

↳ bizonyos use-case renderelődés ⇒ azokat az eventeket mutatja amit külső actor vált ki és a sorrendet

- system event: külső input event amit egy actor generál
- system operation: bizonyos rendszer eventre való válasz

Szerződések:

- Analízis fázisban történik
- ~~rendszer működésének~~
- leírja a műveletek rendszerre való hatását
- szerkezete:
  - név
  - hatáskör
  - típus
  - kereszt referencia
  - jegyzetek
  - kivételek
  - output (nem UI)
  - ~~[illegible]~~ előfeltétel
  - utó feltétel

- Megírás menete:
  1. Rendszerműveletek meghat. a rsz szekv. diagramjáról, minden rendszer-műveletre, szerződés konstruálása
  2. [illegible] Kötelezettségek rész megírása
  3. Utó-feltételek rész befejezése
     - példányosítás és törlés
     - tulajdonság módosítása
     - asszoc. létrehozása, törlése
  4. Előfeltételek meghatározása
     [illegible]
     - fontos, futás közben tesztelendő dolgok
     - dolgok amiket nem lesznek tesztelve, de a művelet sikeressége függ tőle

State diagram

↳ use-case
↳ típusok & szoftver osztályok

Állapotfüggő típusok és osztályok:

- System
- Window
- Application koordinátor (Java applet)
- controller (event handler)
- transaction
- device
- mutator (olyan típus ami megváltoztatja saját típusát v. szerepét)

Interaction diagram - minden műveletnek új

1. diagram amiben ő a kezdő üzenet
2. ha a diagram komplex, osszd kisebb részekre
3. működési szerz + utókövetelmények + use-case ⇒ interaktív objektumokból teremt rendszert

XI V&V: verification and validation

- Verification: jól készítjük el a termeket?
- Validation: a jó terméket készítjük el?

- Szoftver review (statikus rendszer reprezentáció hibák felfedezésére) — statikus
  - doc
  - könyv
  - fejlesztés bármely szakaszában
  - specifikációval való összehasonlítás
  - nem funkcionális tulajdonságokat nem ellenőriz
- Szoftver tesztelés (termék viselkedésének megfigyelése) — dinamikus
  - tesztelési adatokat → viselkedését figyelik

Review: folyamat alapos vizsgálata működtetés nélkül
Review item: amit épp review-zunk
Audit: ellenőrzik hogy a termék minden specifikációnak, követelményeknek stb. megfelel
  egytől független ember végzi
Walkthrough: a fejlesztő bemutatja a review itemet a vizsgáló csapatnak, akik kommentelik a terméket, a termékre fókuszálva (nem a fejlesztőre)
  informálisabb
Inspection: a vizsgáló csapat előre írásait szempontok alapján vizsgálja a terméket, a fejlesztő megfigyelőként van jelen
  formálisabb a walkthrough-nál

Review:
- Meeting előtt:
  - eldönteni mit review-zunk és előkészítjük azt
  - ki vegyen részt? (min 3)
  - ki mit fog csinálni?
    - ↳ koordinátor: szerző, vagy csapat vezetője, meeting levezénylése, keretben tartása (ne térjen el a tárgytól), nyomonkövetés
    - ↳ lead reader: a vizsgálat vezetője
    - ↳ scribe: minden fontosat leír, majd közzé tesz
    - ↳ user representative: user szószólója
  - meeting meghirdetése, anyagok közzététele
- Meetingen:
  - 15 - 2 h
  - cél: azonosítás, nem javítás
  - koordinátor vezényli a meetinget
  - lead reader végigvezet mindenkit a review-kon
  - minden hibát feljegyezni
    - ↳ ki?
    - ↳ hogyan kell adja meg?
    - ↳ milyen a hiba? (súly, típus)
  - a résztvevők eldöntik mit kell tenni
    - ↳ elfogadjuk
    - ↳ elfogadjuk kis változtatás után
    - ↳ elfogadjuk sok változtatás után

- Meeting után:
  - szóbe kézzé tenni a feljegyzéseket
  - a felelősök megteszik amit kell
  - kis változásnál a változást jóvá kell hagyni
  - nagy változásnál új review

Review Report = dátum + team neve + termék neve + review item(s) + részvevők és szerepük + célok + akció elemek (bug, deficiencies, stb)
↳ felelős?
↳ mikorra kell?
↳ súlyosság és típus

Review szabályok:
- felkészülten érkezz
- szerepednek megfelelően cselekedj
- legyél konstruktív és objektív
- ne késs, ne légy el!
- megfelelően vegyél részt a meeting utáni feladatokban
- fogadd el a koordinátor és a csapat felhatalmazását

Szoftver tesztelés

- rendszer működésének tesztelése
  ↳ hibák felfedésére
  ↳ megfelel-e a specifikációnak?
- célja:
  ↳ objektumok interakcióinak ellenőrzése
  ↳ komponensek jól vannak-e integrálva
  ↳ követelményeket jól teljesíti
  ↳ kiadás előtt megtalálni a hibákat
- nem tudja megmondani hogy jól működik a SW (csak ha hiba van)
- nehéz
- 30-50% fejl. költség
- nem-funkcionális elemek ellenőrzésének egyetlen módja
- statikus verifikációval párban kell használni a V&V megtegyen

Unit test:
- programozó
- kis végrehajtható elem
  ↳ OO-ban legkisebb a class
  ↳ lehet class, cluster v. egy végrehajtható
- magában, semmivel se interakcióban
- intuitív

Integration test:
- tapasztalt tesztelő csapat
- egy teljes rendszer vagy alrendszer
- egységek egymástól függnek
- egys. együtt működése
- interfészek az alrendszerben
- Top-down
- Bottom-up

System test:
- független csapat
- teljes integrált alkalm.
- fel. amik csak a teljes rendszerben vannak jelen

Acceptance test:
- formal: systemtest egy részét az end-user vagy kiválasztott emberei vagy a cég emberei az end-user képviseletében újra
- informal (alpha): end-user által, nem szabályszerű tesztelés, de dokumentálva van
- beta: egyedüli tesztelő → saját nézőpontjai alapján tesztel (menedzsment minimálisan részt hat bele)

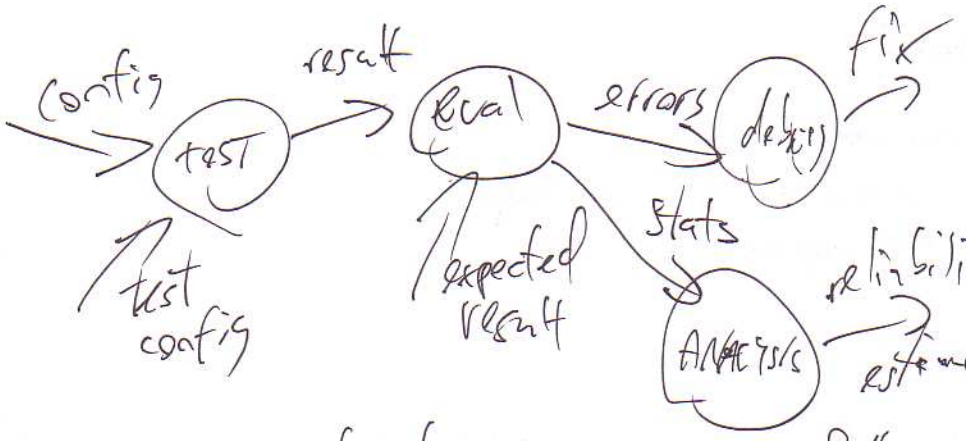
Validation & Verification

Review < Inspection
Walkthrough

fontos

Error – emberi hiba
Fault (bug) – hiba a programban
Failure – program sikertelensége

Test



* Unit – egyes elemek
Integration – összetevők együttműködése
System – az egész megy
Acceptance – megfelel az elvárásnak
Bottom up → testbed
Top down → stubs

formal alpha beta

Minőség dimenziók

FURPS

fontos

Function – elvégzi, biztonságos, célirányos, nagyot
Usability – szép, értelmes, használható
Reliability – nem omlik össze, kapcsolatok rendben
Performance – teljesség, hatékonyság, elegendő, terhelhetőség
Supportability – karbantartható, installációs prob, stb

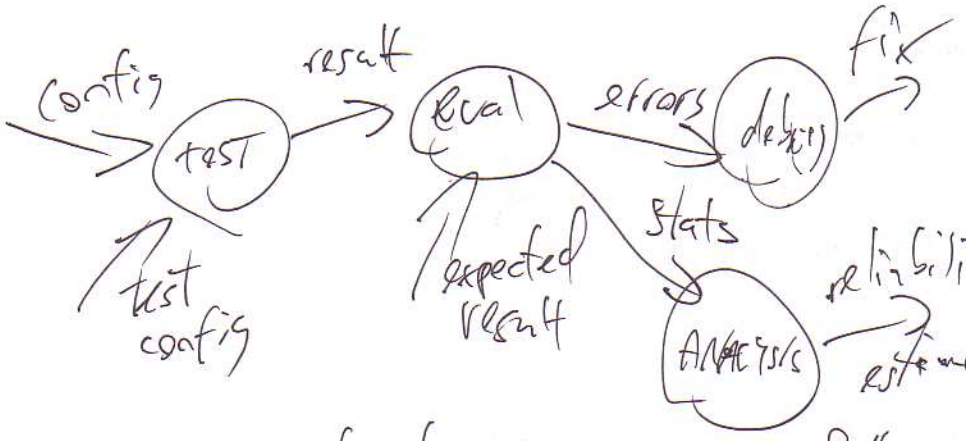
Validation & Verification

Review < Inspection
Walkthrough

fontos

Error – emberi hiba
Fault (bug) – hiba a programban
Failure – program sikertelensége

Test



* Unit – egyes elemek
Integration – összetevők együttműködése
System – az egész megy
Acceptance – megfelel az elvárásnak
Bottom up → testbed
Top down → stubs

formal alpha beta

Minőség dimenziók

FURPS

fontos

Function – elvégzi, biztonságos, célirányos, nagyot
Usability – szép, értelmes, használható
Reliability – nem omlik össze, kapcsolatok rendben
Performance – teljesség, hatékonyság, elegendő, terhelhetőség
Supportability – karbantartható, installációs prob, stb

XII CM – Configuration Management

Fő szempontjai:
- Azonosítás
- Menedzselés
- Státus elnézés & vizsgálat

Fő folyamatai:
- Storage Config. Items
- Változás menedzselés
- Felépítés, építhezet menedzs.
- Kiadás menedzselés

CM: szabály bonyolult rendszerek fejlődésének kontrollálására
↳ szoftver CM: CM szoftverekre specializálva

Azonosítás (Identification)

- config. elemek és configok meghatározása:
  - ↳ config elem: bármi ami információt hordoz
  - ↳ kommunikáció: email, címlista, IRC log, memo
  - ↳ dokumentáció: projekt terv, dokumentációk, user manual
  - ↳ implementáció: fordítók, könyvtár, eszközök, futtatható fájlok
- config: config elemek egy csoportja, mely a project egy bizonyos állapotát jelöli

Menedzsment

- folyamatok és minőségi óvintézkedések definiálása a projekt megfelelő haladása érdekében
  - config item storage menedzs: Hol tárolják az elemeket?
  - változások menedzs: Ki hagyja jóvá?
    Ki implementálja őket?
    Ki ellenőrzi le őket?
  - felépítés menedzs (szoftver gyártás): Miből építhetünk?
    Mi változott a buildben?
  - kiadásmenedzs: Mi lesz a kiadvány?
    Mikor történik kiadás?

Státus leírás és vizsgálat

- projekt státuszának leírása
  - milyen típusú, mennyiségű és súlyosságú hibák vannak és ezek javítva
  - implementált tulajdonságok
  - teljesített követelmények száma
  - megfelelő mértékegységek használatának biztosítása
- vizsgálat (hol tart a folyamat?)
  - a definiált folyamat szerint halad-e
  - próbálják optimalizálni

+ Mikor kell helyzetfelmérés?
- status jegyzés = folyamatos ellenőrzése a projektnek
- audit: a projekt mérföldkövei előtt

Tároló Config Elemek

- tárolás szükségességének meghatározása
  - Kell version control? → hogy csináljuk?
  - Kell backup? → Hogy csináljuk?
- Hol tároljuk az elemeket?
  - verzió kontroll
  - e-mail & naptár rendszer
  - hálózati meghajtó
  - web szerver
- dilemma: mi legyen a külső cégek által kézített eszközökkel?
  - hogyan integráljuk
  - hol tároljuk
- interfészek:
  - Változtatás M. (hogyan?)
  - Építés M. (Mi legyen a temp. fájlokkal?)
  - Kiadás M. (Hol tároljuk?)

Változtatás Menedzsment

- minden változik
  - követelmény
  - hardver kiütés
  - sz.v. kikötés
- kontroll: folyamat definiálása
  - változtatást azonosítani, teljesíteni
  - változtatást végigkövetni (állapotgép)
  - a valódi változásnak legyen köze a változtatási kérelemhez
  - Change Control Board
- változtatás a projekt alakulásával arányban
  - vált. kérelmek feldolgozása periodikusan szünettel - stabilitás
  - új kérelem csak stabilizálás után
  - ha kell, kérelem visszautasítása
- hiba menedzsment
  - hibát javítani
  - ki dönti el a súlyát?
  - Mikor adjuk ki a javítást?

CR (Change request): formális termék, ami tartalmaz minden érintett követelményt (új ficsör, új request stb) a hozzá tartozó státus információkkal a projekt egészére kiterjedően minden itt van eltárolva (dátum, indoklás, stb) és a projekt végéig mindig elérhető.

CCB (Change Control Board): tábla a változtatás átlátására (rajta van hogy ki mindenkinek mi köze van a változáshoz)

Felépítés menedzsment

- baseline: pillanatfelvétel egy bizonyos időben minden egyes elemről
  egy bizonyos verziójáról a projektben
- delta: egy db baseline-tól a következőig történt változások
- Miért?
  - ↳ megismételhető
  - ↳ követhető
  - ↳ ellenőrizhető

Kiadás menedzsment

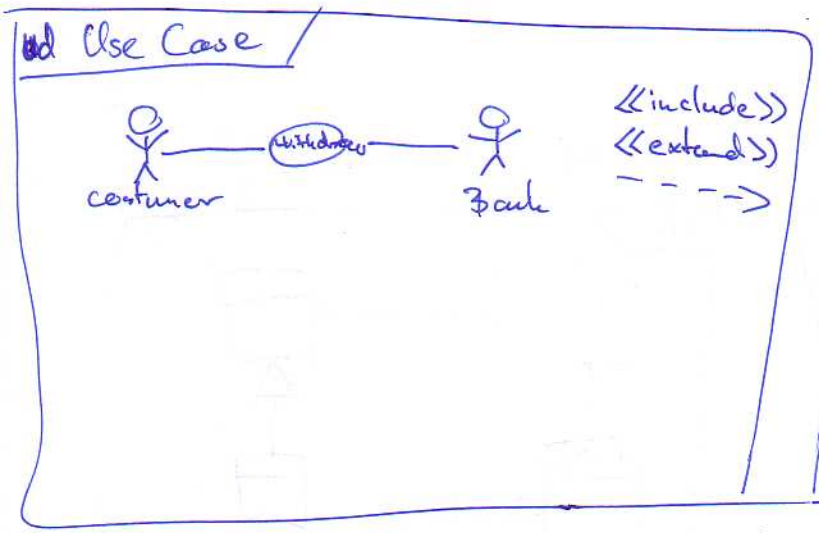
- definíciók:
  - verzió: funkcionálisan eltérő változat (egy számmal)
  - variáns: funkcionálisan egyezik, nem-funkcionálisan eltér
  - kiadás: olyan verzió, amit a dev. team-en kívül is publikálnak
  - javított kiadás: verzió + variáns (de nem új release!)

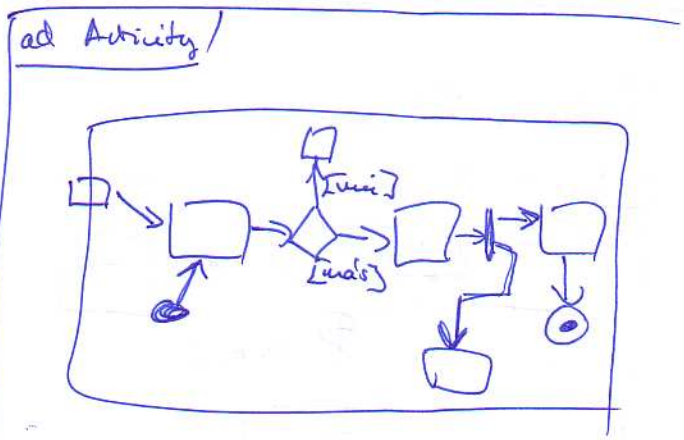
- verzió névproblémája: Build 1234 vagy Ultimate Edition XP SP2 ?

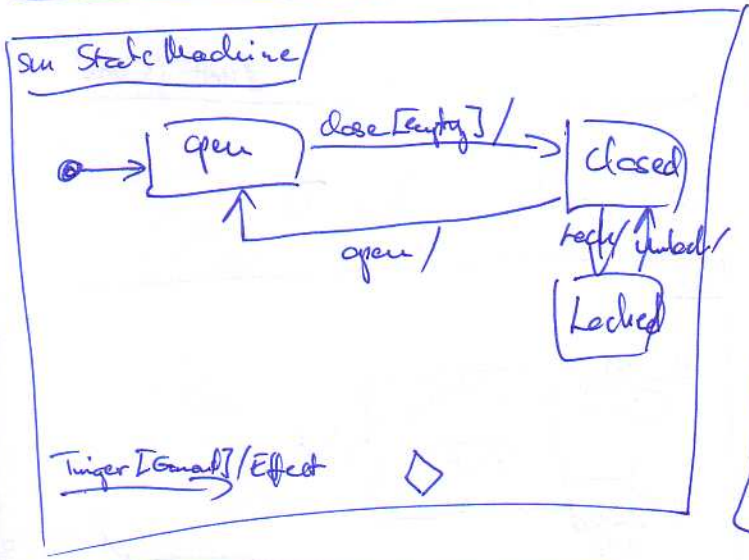
→ verziókat számozzuk
  - ↳ folyamatosság-orientáltan
  - ↳ változtatás-orientáltan

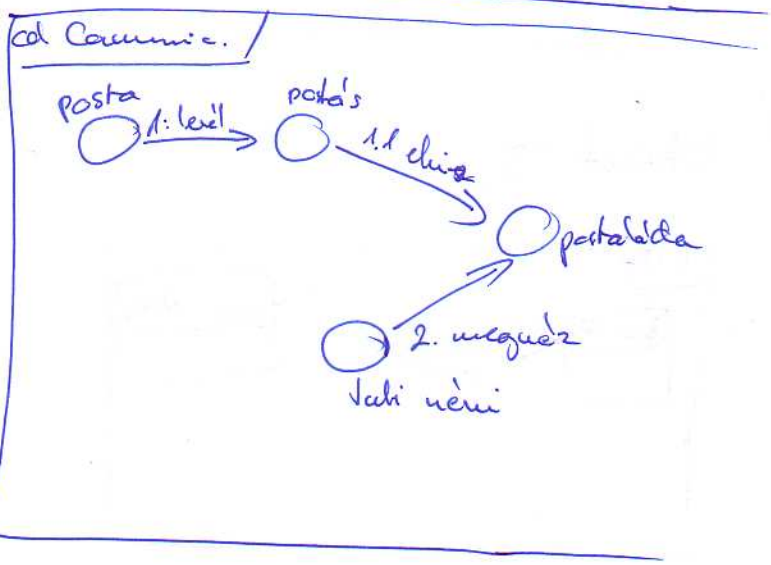
- check-out: adott elemet szerkesztésre visszatartani
- check-in: változtatott elemet publikálni
  - ↳ SVN – változat, update-el, merge-el

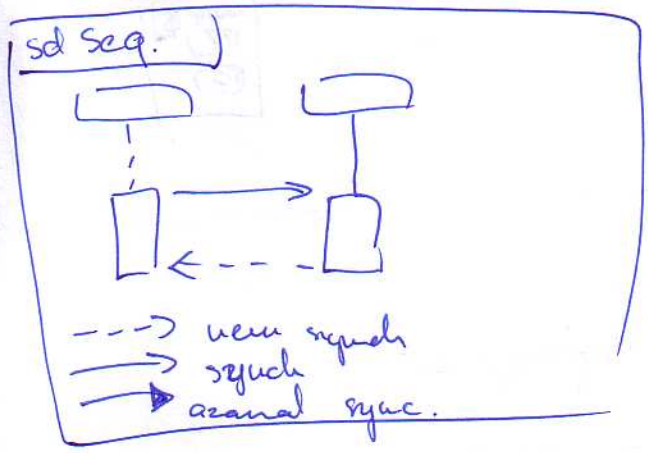
Behavioural Models

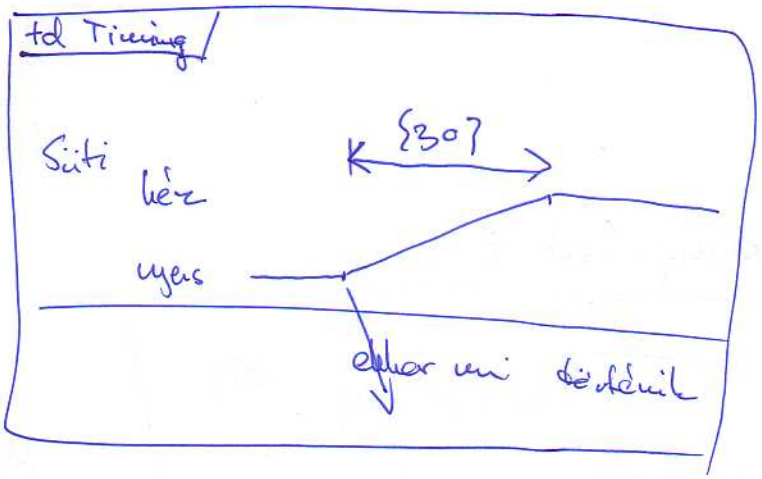








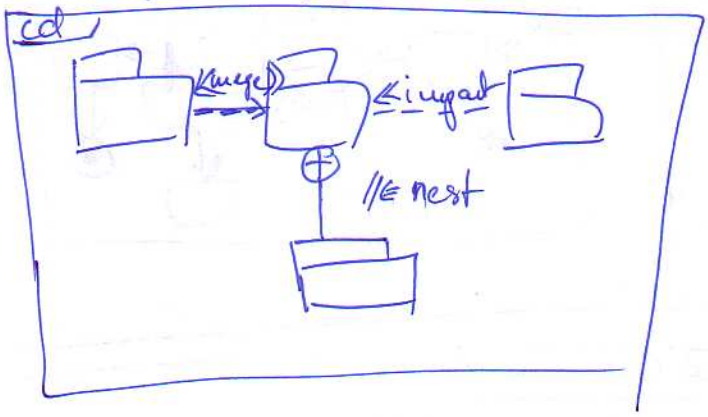




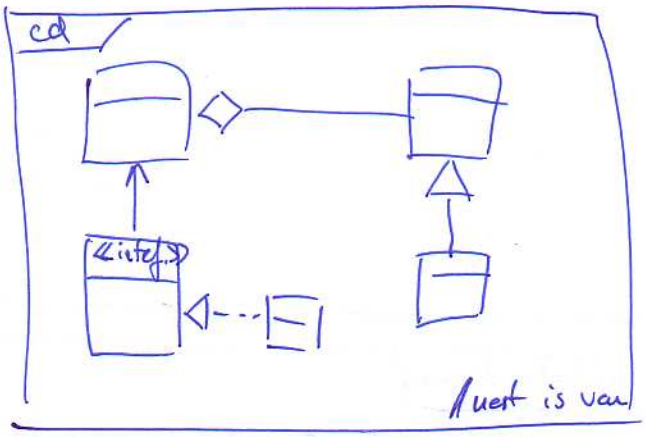
UML 2

Strukturmodelle

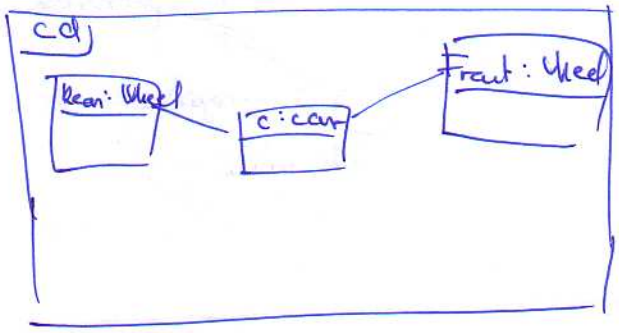
Package Diagram



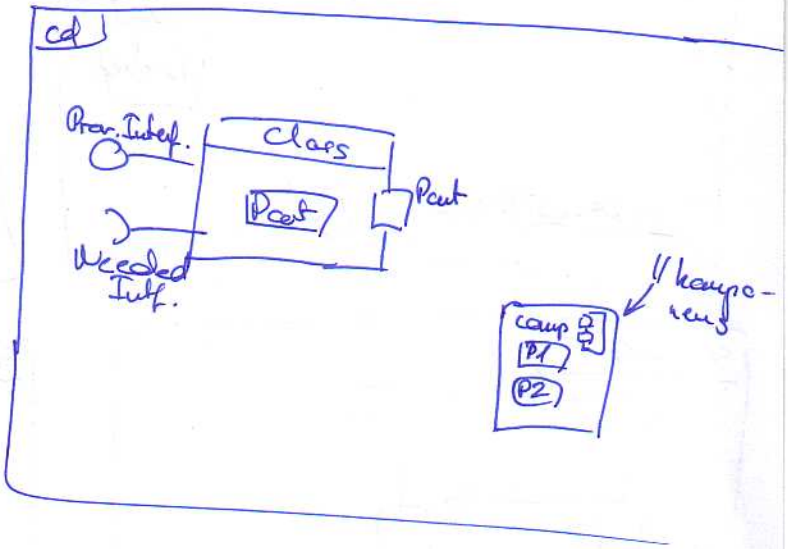
Class Diagram



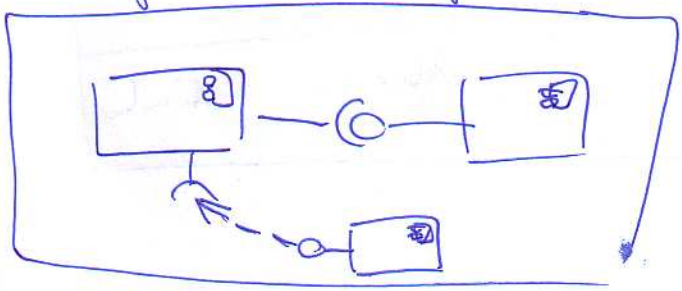
Object D.



Composite Str. D



Component D Diagr.



Deployment