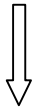
I. Bevezetés

Mi az a szoftver?

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



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

Szoftverfejlesztés problémái:

- kereslet magasabb, mint a kínálat
- emelkedő minőségi elvárások
    - helyesség (azt csinálja amit kell, ahogy le van írva a dokumentációban)
    - megbízhatóság (nem omlik össze különleges esetekben sem)
    - teljesítmény (sebesség, hardverigény stb.)
    - hordozhatóság
- visszamaradt (legacy) szoftverek megkerülhetetlensége
- régi szoftver újrafelhasználásának szüksége

Szoftverfejlesztés céljai:

- programírás hatékonyságának növelése
- szoftver minőség javítása
- olcsóbb fejlesztési és fenntartási költségek elérése
- fejlesztő szoftverek előállítása
- pontosabb költség és kockázatfelmérés támogatása
- szoftverfejlesztési folyamat automatizálása

Mérnöki munka:

- költséghatékony megoldások előállítása
- gyakorlati problémákra
- tudományos ismeretek alkalmazásával
- építve dolgokat
- az emberiség szolgálatában

szakma + termelés = kereskedelmi termelés
kereskedelmi termelés + tudomány = mérnöki tervezés

Szoftverfejlesztés kudarca:

- lemaradás ütemtervtől
- becslést átlépő költségek
- megbízhatatlanság
- fenntarthatósági problémák
- gyenge teljesítmény

Okok:

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

Szoftvertervezés:

- menedzsment munka
- tudományos
- technikai
- jogi
- gazdasági
- technológiai

People	Emberek
Problem	Probléma
Process	Folyamat
Product	Termék

II. Szoftverfejlesztési folyamat

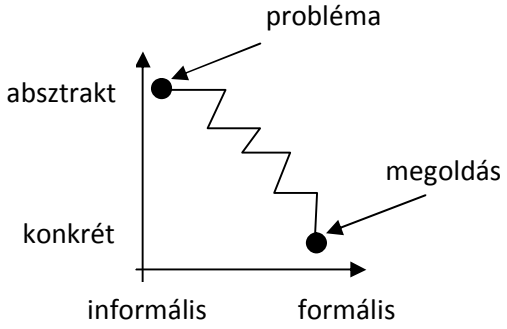
Minőség: Egy termék jellemzőinek és tulajdonságainak összessége, amely meghatározza a program azon képességét, hogy megadott igényeket kielégítsen.
Ezek lehetnek:

- megfoghatatlan minőségi jellemzők
- felhasználói minőség
- előállítási minőség
- termék minőség
- ár-érték arány

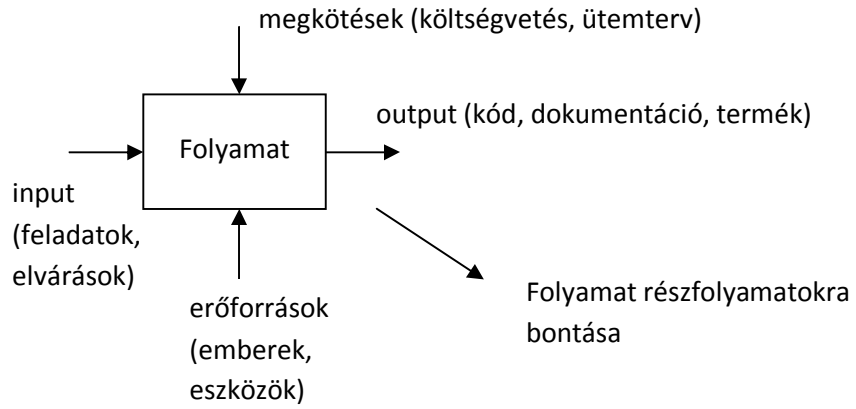
Folyamat: nyersanyagok (input) kész, felhasználóknak hasznos termékké (output) alakítása emberek által ismeretek, módszerek, eszközök, felszerelés felhasználásával.
Folyamatmenedzsment elve: Egy termék minősége arányos az őt előállító folyamat minőségével.
Fejlesztés, javítás:

- jobb minőség
- nagyobb hatékonyság
- olcsóbb
- rugalmasabb
- előállítói elégedettség növelése

Absztrakt vs. konkrét: Mennyire magával a problémával, illetve annak egy elvonatkoztatott formájával foglalkozunk.
Formális vs. informális: Mennyire kötött, egyértelmű módon fogalmazzuk meg a megoldási lehetőséget.
Elméletben:



Folyamatmodellezés:



Termék életciklusa:

I. Igénymeghatározás és felmérés (Requirement)
- mi a probléma?
- mik a 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?

IV. Megvalósítás és építkezés (Implementation & Construction)
- megtervezett lépések végrehajtása

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

VI. Fenntartás, karbantartás (Maintenance)
- milyen fejlesztések, változtatások szükségesek?

Fázisos modellek

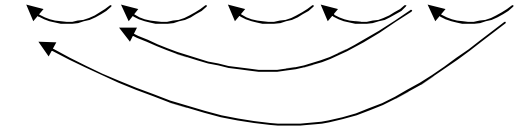
Lineáris modell:

Req → Spec →Design →Impl →Valid →Maint

Folyamatosan növekvő hibajavítási költségek, könnyen maradnak tervezési hibák is

Vízesés modell:

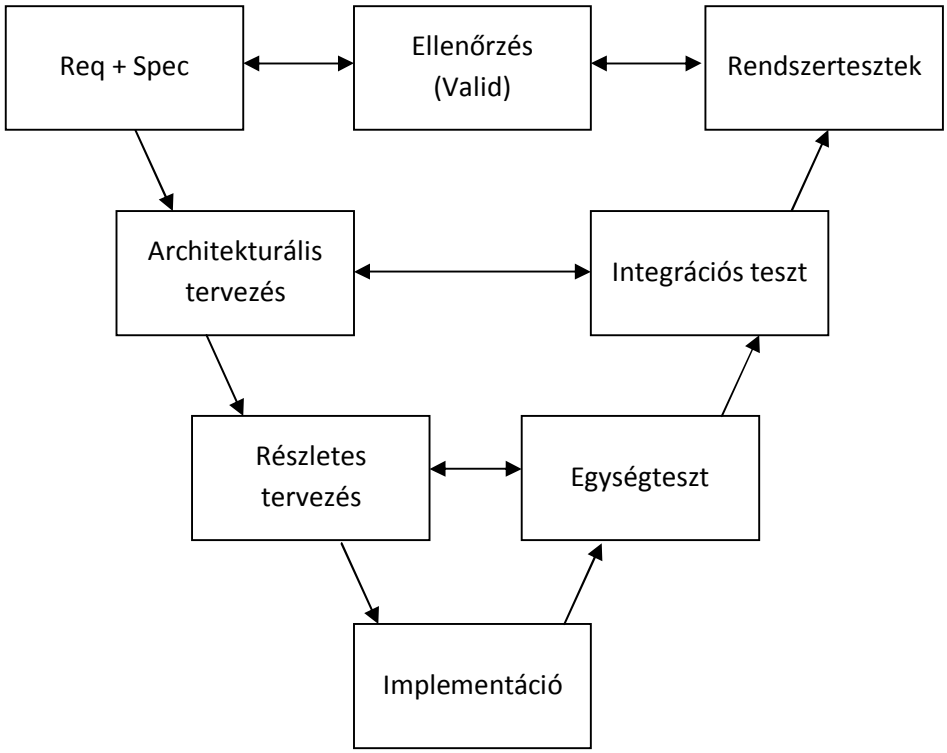
Req → Spec →Design →Impl →Valid →Maint



Átadható teljesítések:

- Req: rendszerdefiníció, megvalósíthatósági jelentés
- Spec: megoldásleírás, elvárásdefiníció, előzetes felhasználói dokumentáció
- Design: architektúra terv, felhasználói dokumentáció, ellenőrzési terv
- Impl: fejlesztői dokumentáció, elfogadhatósági vizsgálat terv
- Valid: teszteredmények, fejlesztés tervek, projektértékelés

V modell:



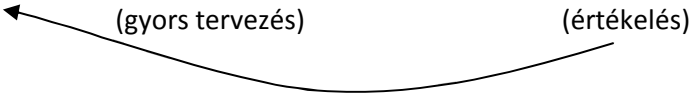
Fázisos életciklus problémái:

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

Iteratív modell:

Req → Spec →Quick Design →Implementation →Evaluation

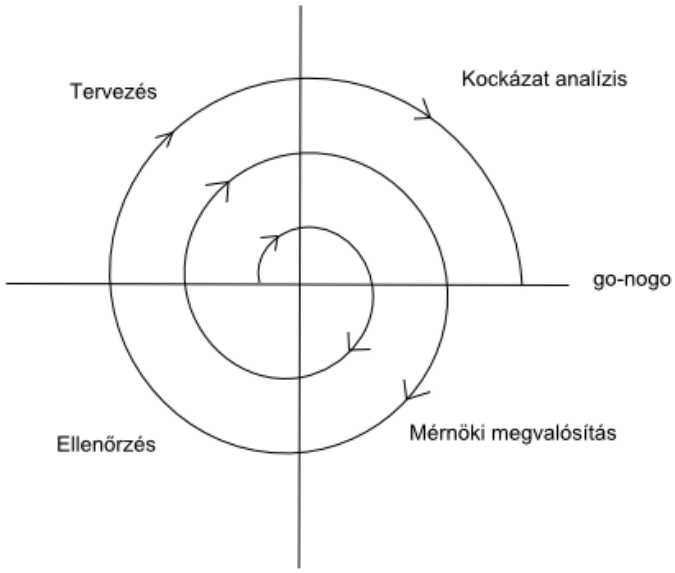
(gyors tervezés) (értékelés)



RAD modell: (Gyors alkalmazás fejlesztés)

- üzleti modell (mi kell?)
- data modell (mivel oldható meg?)
- process modell (hogyan csinálják?)
- application (előállítás)
- testing (ellenőrzés)

Spirális modell:



Capability Maturity Model

CMM

Minőségi tervezési folyamat mérésére alkalmas

Szervezési fejlesztéseket tartalmaz

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

5 szint

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

Kezdeti:

- ötletszerű fejlesztés, gyakran kaotikus
- eredmény egyéni teljesítménytől függ
- nincs menedzselési procedúra
- nincs projektterv
- nem ellenőrzött

Megismételhető:

- a projekt sikeresen megismételhető
- informális (nincs formális fejlesztési modell)
- követelmények menedzselve (adminisztráció, felülvizsgálat, változtatás)
- menedzselt és felügyelt projektterv

Meghatározott:

- definiált és dokumentált szoftverfejlesztési folyamat → javítás lehetősége
- a projektek ugyanazokhoz a sztenderdekhez igazodnak
- a fejlesztés lépései részfolyamatokra vannak bontva, ezek struktúrája ismert és kontrollált

Szabályozott:

- a folyamat és a termék folyamatos minőségellenőrzése megoldott
- a folyamat minden részegysége meghatározott, mérhető és ellenőrizhető
- a problémák a létrejöttük helyén elérhetőek
- a haladás pontosan ellenőrizhető
- formális program

Optimalizáló:

- a részletes és folyamatos minőség-ellenőrzés eredményei lehetővé teszik a folyamat fejlesztését
- új ötletek és megvalósítások ellenőrzötten építhetők be
- a javítást elősegítő lehetőségek folyamatos kipróbálás alatt állnak

III. Projektmenedzsment

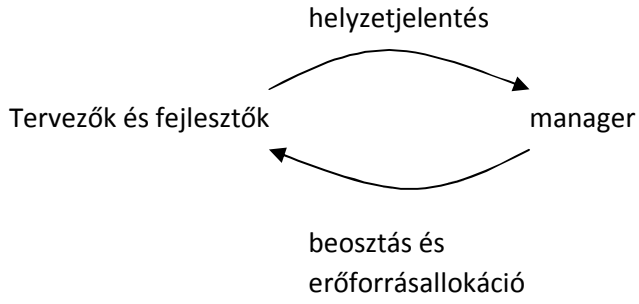
Csoportszervezés

Alapelvek:

- kevesebb, jobb ember használata
- személyhez szabott feladatok kiosztása
- hosszú távon megéri kihozni a legtöbbet
- megéri harmonikus, jól együttműködő csapatot összeállítani
- aki nem való a csapatba, azt ki kell dobni

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

Tervezés:



erőforrások:

- ember
- hardvereszköz
- szoftver segédeszköz

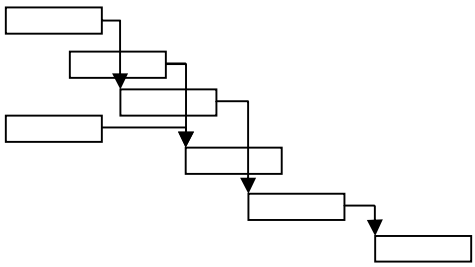
szabályozók:

- időigény
- információ (pl. dokumentáció)
- minőség
- pénz

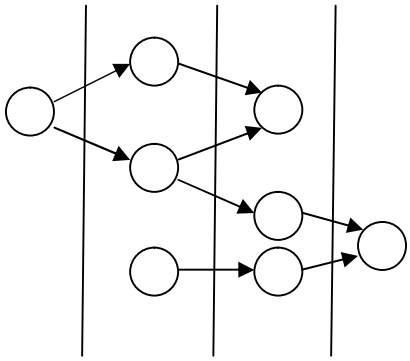
- folyamat részfeladatokra bontása a függőségek megállapítására
- párhuzamosítás
- erőforrásallokáció

Pert vagy Gantt diagram

Gantt:



Pert:



Kockázatanalízis

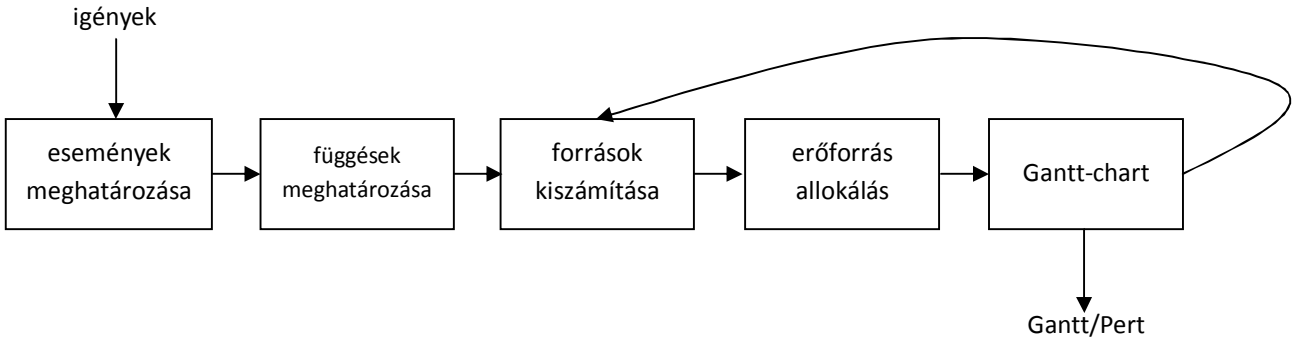
Kockázat: Annak a valószínűsége, hogy valami váratlan körülmény negatívan hat a projektre.

- projekt kockázat → idő vagy erőforrás beosztását befolyásolják
- termék kockázat → a szoftver minőségét vagy teljesítményét befolyásolja
- üzleti kockázat → a szoftverfejlesztő cégre ható piaci jelenségek

Kockázatmenedzsment:

- Azonosítás (mik)
- Analízis (mekkora eséllyel, milyen súlyos)
- Tervezés (hatás elkerülés és minimalizálás)
- Megfigyelés (fellépő kockázatok megfigyelése)

Időbeosztás készítése



IV. Probléma-meghatározás (Requirement)

Minél később találnak meg egy hibát, annál drágább kijavítani.
└►hibák nagyobbik rész probléma-meghatározás

Elvárás definíciók: rendszerleírás a felhasználóknak
Elvárás specifikációk: megállapodási alap a vásárló és a készítő között

Viselkedési elvárás: mit csinál
Nem viselkedési elvárás: milyen tulajdonságokkal csinál valamit

Elvárások: természetes nyelven, fölösleg nélkül, strukturált formában, ellenőrizhető állítások formájában.

- helyes (minden elvárás kell)
- egyértelmű
- teljes (minden ami kell ott van)
- ellenőrizhető
- érthető
- módosítható
- nyomon követhető (a megoldás menetében láthatóvá tehetők az elvárás elemei)

Irányelvek (Principles):

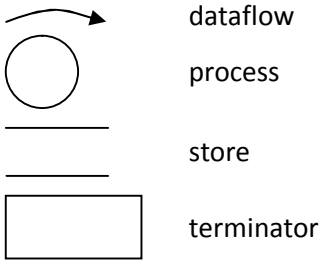
- információtartalom: adat és vezérlő objektumok, amik a lehetséges inputokat adják a rendszernek
- információáramlás: adat és vezérlés változása a rendszerben
- információszerkezet:
    - miket kell a rendszernek megcsinálnia?
    - hogyan viselkedik a szoftver?

Információ capture

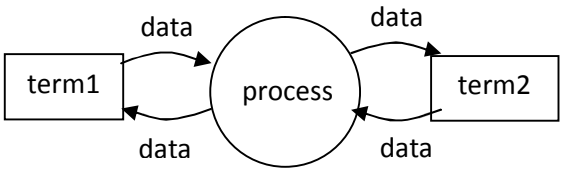
- data – dataflow
- process – function
- control – control flow
- kommunikáció
- konkurencia
- időzítés – szinkronizáció
- kapcsolatok
- valódi
- history

V. Specifikáció

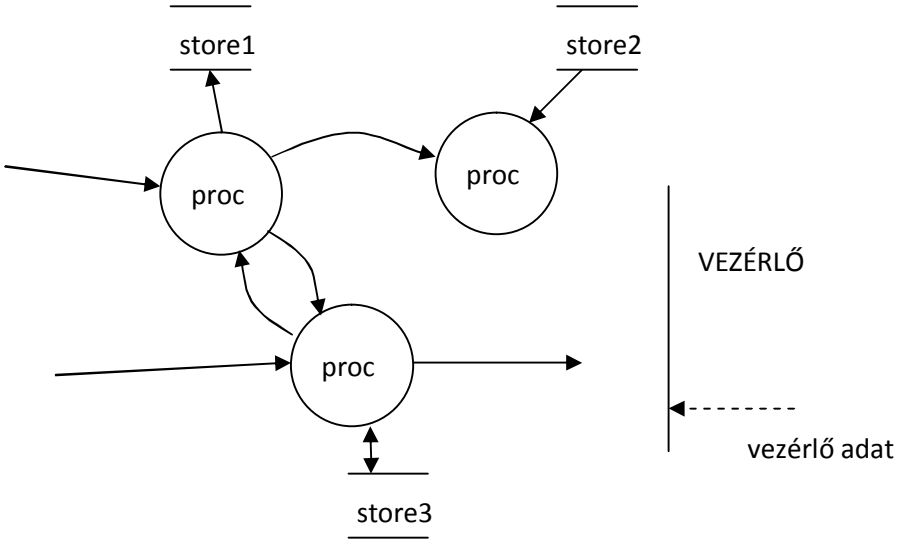
DFD



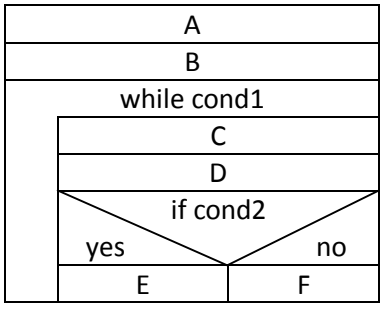
- *Context-diagram*



- *Internal process*



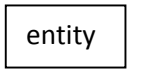
Nassi-Schneiderman diagram



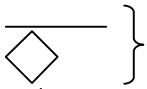
pszeudokódban:

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

Entitás relációs diagram



entitás/adat



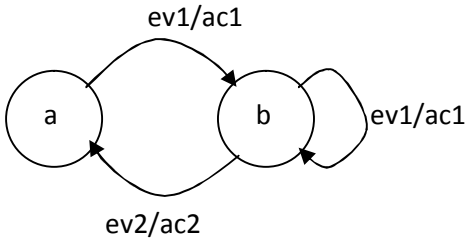
kapcsolat

(több is) kardinalitás

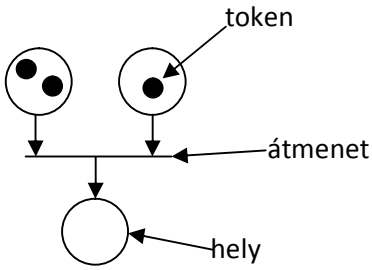
modalitás (van v nincs)

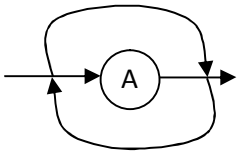
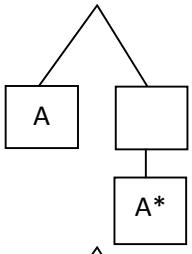
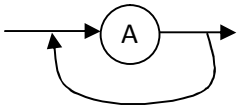
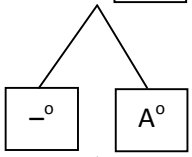
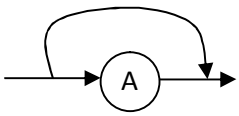
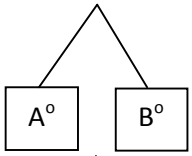
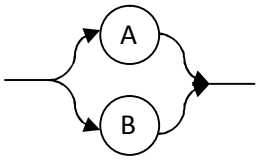
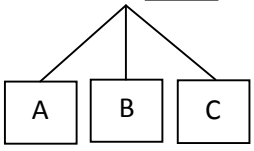
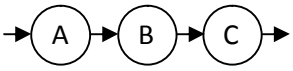
Behavioral modeling

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



Petri-háló



	Algebrai definíció	DTD	BNF	JSD	Szintaxis gráf
0…n	{A}	A*	{<A>}		
1…n	A + {A}	A+	<A>{<A>}		
0…1	0{A}1	A?	[<A>]		
VAGY	[A|B]	(A|B)	<A>|<B>		
SEQ	+	,	space		

VI. Tervezés

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

- belső szerkezet és folyamatok specifikálása
- tervezési döntések feljegyzése, választások megindoklása
- véglegesíteni a tesztelés tervét és a user manualt
- tervrajzot adni az implementációhoz, teszteléshez és karbantartáshoz

Architektúra tervezés:

- a rendszer koncepcióját finomítani
- belső folyamatok meghatározása
- magas szintű folyamatok → részfolyamatokká
- belső adatfolyam és feldolgozás
- függvények közti kapcsolatok és kapcsolódási pontok meghatározása
- adatfolyam
- adattár

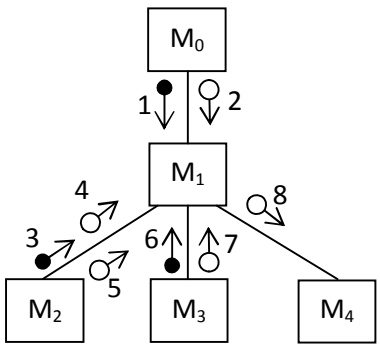
Részletes tervezés:

- algoritmusok specifikációja, amik függvényeket implementálnak
- adatstruktúrák, amik adattárakat valósítanak meg
- valódi kapcsolat az adat és a függvények között

Szempontok:

- ∅ csőlátás
- nyomon követhető, analizálható
- a kereket már feltalálták, köszi
- problémával foglalkozni
- facilitate change
- számítsunk a váratlanra!
- nem kód!
- minősége van
- review

Modularity-Structure diagram
pl:



Coupling (csatolás):

- ha már szétszedtük a rendszert modulokra, azok közt relációk léteznek
    - más dekompozíció → más relációk
- a belső kapcsolatok megmutatják egy változás behatását
- fenntarthatóság fontos indikátora
- tervezési minőség fontos tulajdonsága
    - dimenziói:
        - mi kommunikálódik? (adat, stamp, control stb.)
        - kapcsolat mérete
        - kapcsolat ideje

Strukturált programozás

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

pl: JSP

Absztrakció → csak a lényeggel foglalkozik, a többit elhanyagolja, egyszerűsíti a problémát

- kontrollálja a részletezés szintjét és mennyiségét
- 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 valahogy
- vezérlés – nem fejtjük ki, mi a tényleges szerkezete, hogy egy vezérlési folyamat hogyan működik → jelöljük a kívánt hatást

Egységbe zárás (encapsulation): összezár néhány objektumot logikailag vagy egy fizikai tárolóba

- process: ezt csinálom
- entity: egy objekt, amivel ez megtörtént

Dekompozíció: Egy program teljes komplexitása nagyobb, mint a kisebb részegységek (feladatmodulok) és azok összekötésének kompozíciója.

- fan in – meghívó modulok száma
- fan out –meghívott modulok száma
- döntés hatáskör – miről tud önálló döntést hozni?
- irányítási hatáskör – mit tud csinálni?
    - irányítási hatáskör < dontési hatáskör → döntési hasítás

Modularitás → feladatmodulok
Szedjük szét modulokra a nagy feladatokat, hátha a modulokat könnyebb kezelni.

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

erős	Funkcionális	–	ugyanazt csinálják
↑	Szekvenciális	–	egymás után csinálják (output → input)
	Kommunikációs	–	ugyanazon csinálnak mást
	Procedurális	–	kb. egymás után csinálják (output ⇢ input)
	Időbeli (temporális)	–	egyszerre kell csinálni
↓	Logikai	–	van közük egymáshoz (laza kapcsolat)
gyenge	Véletlen	–	nincs kapcsolat

Csatolás: Mit kommunikál két modul egymással

- data (adat)
- stamp (összetett adat)
- control (modulvezérlés)
- common (közös adategységek)
- contest (közös kód)

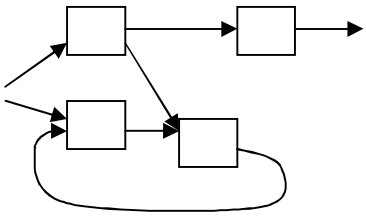
Jackson Structured Programming – JSP

1. I/O adatok definíciója
2. struktúrák felbontása
3. futtatási információk előállítása
4. kódtranszformáció

Szoftverarchitektúrák

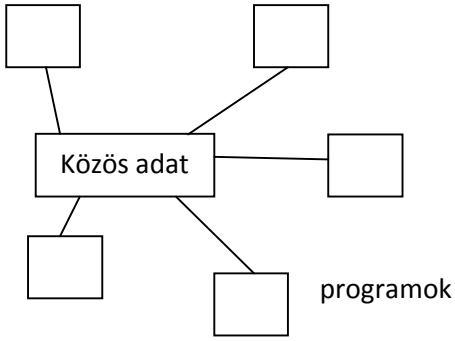
I. Adatfolyam
   - Batch (Output → Input → Output → stb.)
   - Cső/szűrő (Pipe & Filter) – felosztja részfeladatokra majd azok outputját kötögeti

     kapcsolat adattranszformáció műveletek
     +újrahasznosítható
     +párhuzamos
     -nem iteratív
     -túlságosan egymás utáni lépések

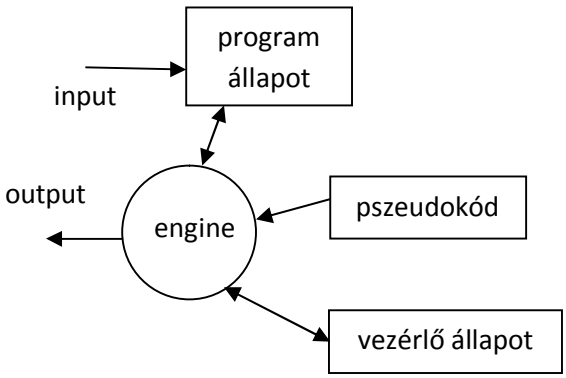


II. Hívásos rendszerek
   - Szubrutinos rendszerek
   - OO rendszerek
   - Rétegelt rendszerek → lásd később

III. Adat alapú rendszerek
   - Adatbázisok
   - Hypertext (linkelt tartalom)
   - Blackboard
     - önálló részegységek
     - bonyolult, gyenge



IV. Virtuális gépek
    - fordítórendszer (interpreter)



Rétegzett architektúrák

Vízszintes: Alulról várják, felülre készítenek

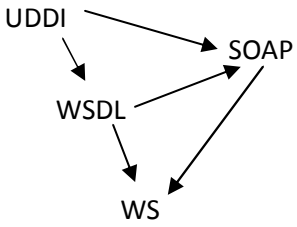
Függőleges: Azonos feladatokat egymás alá

Kliens-szerver architektúra:

- GUI
- BOM (Business Object Model)
- DB (Database)

SOA – Service Oriented Architecture:

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



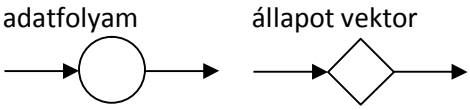
VII. JSD

Modell:

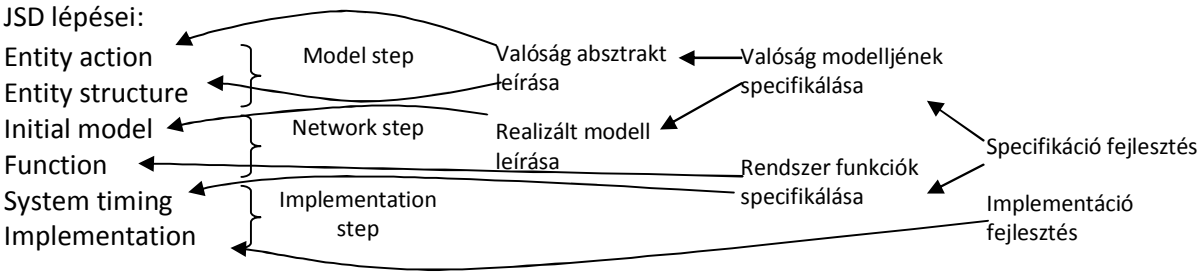
- alkalmazási terület = entitások csoportja
- *Entitás*: a világ egy létező eleme, ami eseményt generál vagy szenved el
- Modeling = entitások felfedezése
- modellezés, mint funkciók környezete
- a modell stabilabb, mint a függvények
- fenntarthatóság

Folyamat:

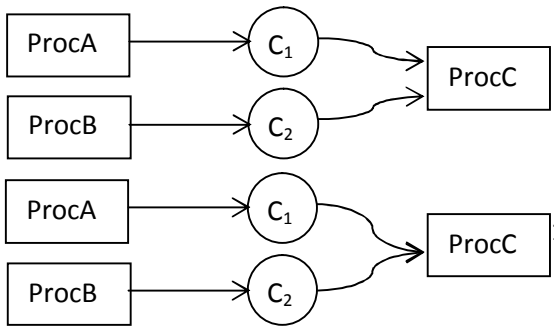
- folyamat példány:
  - program szöveg
  - állapot vektor (test pointer)
- folyamatok kapcsolata:



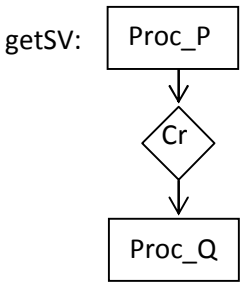
- folyamat implementáció
  - specifikáció közvetlen implementálása
  - tervezés ütemező
  - állapot vektor szeparálása
  - program szöveg feldarabolása



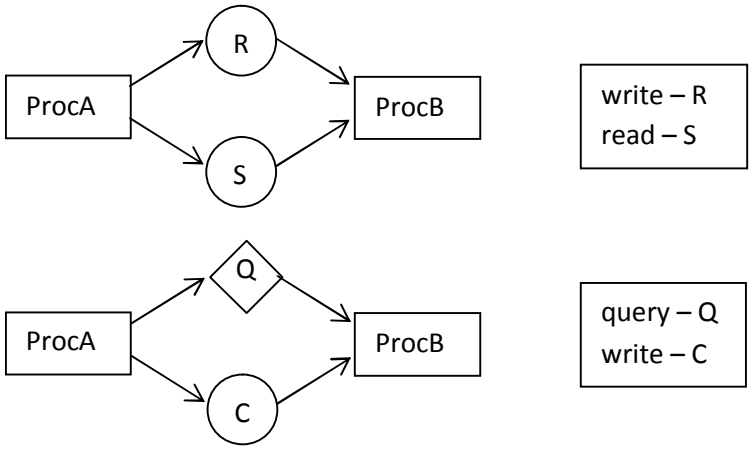
1. Entity action – entitások és események azonosítása
   - Entity? → főnevek, dolgok, szerepek stb.
   - Action? → igék, elemi dolgok, külső világból
2. Entity structure
   - entity life history alapján processz-hálózat felvétele
   - közös történések
   - Jackson Structure Diagram az entitásokra
3. Initial model
   - System Specification Diagram (SSD)
   - adat-folyam kapcsolatok
     - végtelen FIFO
     - pl: read (suspend), write



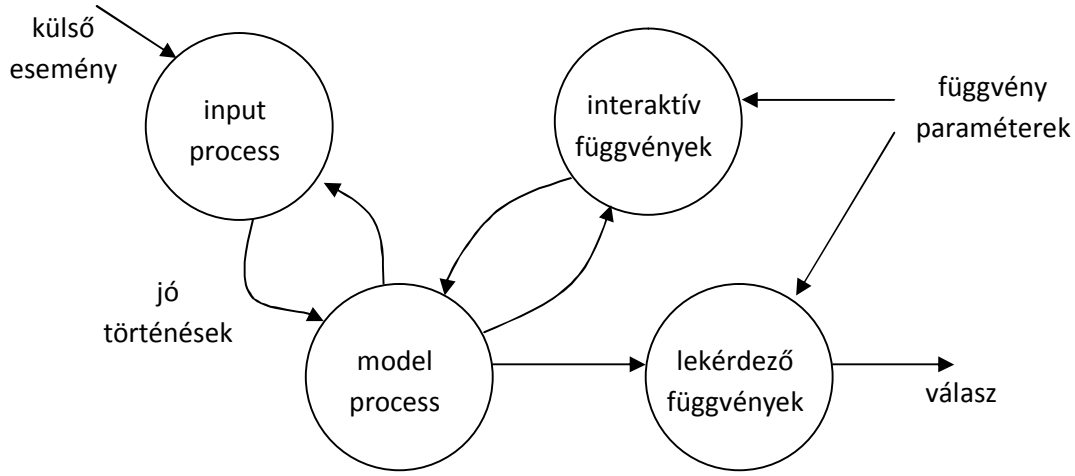
- state vector kapcsolódás
    - pl: proc_Q indítja a folyamatot



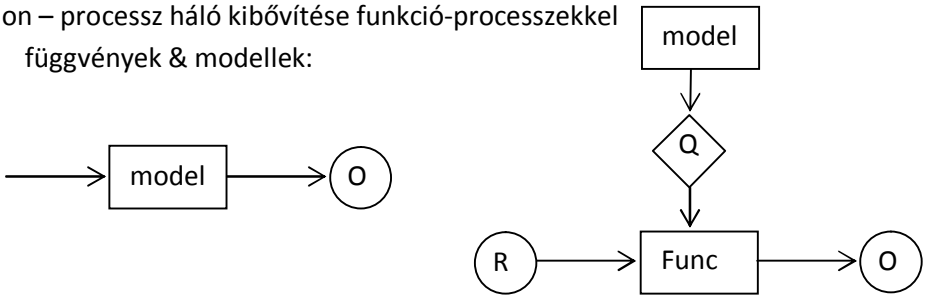
- megkötések



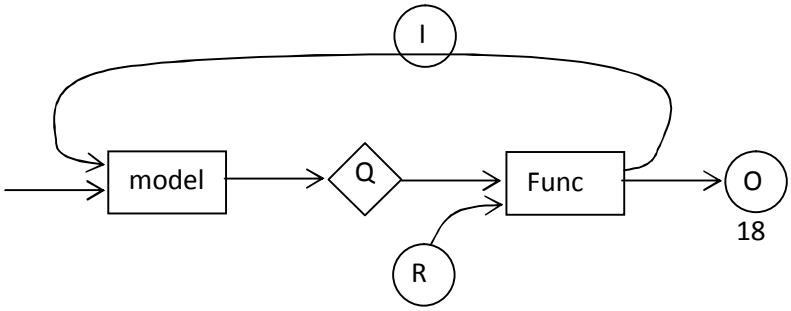
- való világ + modell + függvény kapcsolódások:



4. Function – processz háló kibővítése funkció-processzekkel
    - függvények & modellek:



    - iteratív függvények:



5. System Timing
    - kimenetek megkötései
    - getSV gyakorisága
    - model-process frissítési frekvencia
    - time-grain marker olvasása
    - hazárdok
6. Implementation
    - processzek ütemezése
    - inverziók

VIII. RUP – Rational Unified Process

- lépcsőzetes és iteratív
    - több ciklusos analízis, tervezés és építés
    - korai feedback a megrendelőnek és developernek
    - fokozatosan újraalkalmazható eszközök
    - Req. változtatható → piaci megfelelés
    - korai átfogó integrációs tesztelés
- Use-case orientált
    - Req. megjelenik a use-case-ben
    - becsléseket a use-case befolyásolja
    - időbeosztások use-case alapján
    - koncepció, osztályok és tesztek use-case alapján
- Architektúra központú
    - korai rendszerarchitektúra készítésre ösztönöz
    - rétegelt architektúra
    - alrendszerek közt kicsi kapcsolat
    - könnyen érthető
    - robusztus, sokat bír, nagy rendszerre is alkalmazható
    - sok újrahasznosítható komponens
    - fontos és kockázatos use-case-ek alapján vezérelt
    - jól definiált interfészek

Struktúrált folyamatok:

- Mikor? → életciklus felosztása fázisokba
- Mi? → jól definiált történések
- Mi készül? → egy bizonyos elemcsoport terméke
- Ki csinálja? → HR

Életciklus fázisai:

- Inception (előkészítés): piaci igény felmérése, projekt behatárolása, projekt elképzelés specifikálása
- Elaboration (kidolgozás): források és tevékenységek meghatározása, jellemzők specifikálása, architektúra tervezés
- Construction (megvalósítás): működő rendszer megépítése és tesztelése az előzetes terveknek megfelelően ismételt lépésekben
- Transition (átadás): terméket átadjuk a user-eknek (legyártás, kiszállítás, training)

Munkafolyamatok (workflow):

Requirements (igényfelmérés)

- vázlatos terv
- előzetes nyomozási jelentés
- igények meghatározása
- megfogalmazások lejegyzése
- prototípus implementálása
- use-case meghatározása (a fontosakat)
- vázlatos fogalmi modell
- vázlatos rendszer architektúra

- Mit kell tudnia a terméknek?
- Ezt olyan formában leírni, hogy a vevő és a developer team is könnyen megértse
- Tartalma:
    - áttekintés
    - vevők
    - célok
    - rendszer funkció
    - rendszer tulajdonságai
    - feltételezések
    - kockázatok
    - szójegyzék

Analysis (analízis)

- elengedhetetlen use-case-ek definiálása
- use-case diagramok finomítása
- fogalmi modell finomítása
- szójegyzék finomítása
- rendszernek szekvencia-diagrammot rajzol
- működési szerződés
- állapot diagram

Design (tervezés)

- valódi use-case-ek
- jelentések meghatározása, UI, storyboardok
- rendszer architektúra meghatározása
- interakció-diagram
- osztálydiagrammok
- adatbázis-séma

Implementation (megvalósítás)

- class & interface implementálás
- metódusok implementálása
- ablakok implementálása
- jelentések implementálása
- adatbázis-séma implementálása
- teszt kód írása

Deployment (telepítés)

- párhuzamos futás & keresztezések megállapítása
- technikai dokumentáció befejezése
- user dokumentáció befejezése
- rendszerteszt
- acceptance test
- document test
- training
- support létrehozása
- telepítés

Use-case-ek

- actor-based
    - egy rendszerhez vagy szervezethez tartozó actorok azonosítása
    - minden actorhoz azonosítani, hogy melyik folyamatot indítja vagy módosítja
- event-based
    - külső események azonosítása, amire a rendszernek válaszolni kell
    - összekapcsolni a megfelelő actorokat, eventeket és use-case-eket

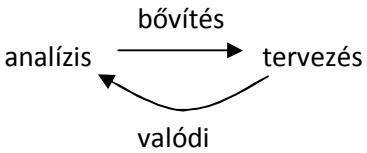
Formái:

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

Use-case készítése

1. rendszer funkciói alapján → rendszerhatások majd actorok és use-case-ek megállapítása
2. magas szintű 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 tervezési folyamatban jönnek elő
7. sorrendezés
    1. az architektúra magjára hatnak
    2. kockázatos, időkritikus, komplex
    3. kutatást igényel, vagy új, kockázatos fejlesztést
    4. elsődleges üzleti vonulatot képvisel
    5. közvetlen support, növekedett bevétel, csökkent költségek

Ami ismételhető:



Fogalmi modell

1. Lehetséges koncepciók listája
2. Maradj a jó tervnél
3. Fogalmi modell rajzolása
4. Koncepciók közti asszociációk megrajzolása
5. Fölösleges vagy helytelen asszociációkat kidobálni
6. Tulajdonságok meghatározása
7. Generelizáció – specializáció hierarchiák
8. Szójegyzék

Gyakori asszociációk:

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

System Sequence Diagram

Bizonyos use-case rendeződés → azokat az eventeket és azok sorrendjét mutatja, amiket külső actor vált ki.

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

Szerződések:

- analízis fázisban történik
- leírja a műveletek rendszerre való hatását
- szerkezete:
    - név
    - hatáskör
    - típus
    - kereszt referencia
    - jegyzetek
    - kivételek
    - output (nem UI)
    - előfeltétel
    - utófeltétel
- megírás menete:
    1. Rendszerműveletek meghatározása a rendszer szekvencia diagramjáról, minden rendszerműveletre szerződés konstruálása.
    2. Kötelezettségek rész megírása.
    3. Utó-feltételek rész befejezése.
        - példányosítás és törlés
        - tulajdonság módosítása
        - asszociáció létrehozása, törlése
    4. Előfeltételek meghatározása
        - fontos, futás közben tesztelendő dolgok
        - dolgok amik 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 vagy szerepét)

Interaction diagram – minden műveletnek új

1. diagram amiben ő a kezdő üzenet
2. ha a diagram komplex, osszuk kisebb részekre
3. működési szerződés + utókövetelmények + use-case → interaktív objektumokból tervezz rendszert

IX. V&V: Verification and validation

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

Szoftver review (statikus rendszer reprezentáció hibák felfedezésére) statikus

- olcsó
- könnyű
- fejlesztés bármely szakaszában
- specifikációval való összehasonlítás
- nem funkcionális tulajdonságokat nem ellenőriz

Szoftver tesztelés (termék viselkedésének megfigyeélse) dinamikus

- tesztelési adatokat kap → viselkedését figyelik

Review: folyamat alapos vizsgálata működtetés nélkül
Review item: amit épp review-olnak
Audit: ellenőrzik, hogy a termék minden specifikációnak, követelménynek stb. megfelelt-e, cégtől független ember végzi
Walkthrough: a fejlesztő bemutatja a review itemet a vizsgáló csoportnak, akik kommentelik a terméket, a termékre fókuszálva (nem a fejlesztőre). Informálisabb.
Inspection: a vizsgáló csapat előre összeírt szempontok alapján vizsgálja a terméket, a fejlesztő megfigyelőként van jelen. Formálisabb a walkthrough-nál.

Review

- Meeting előtt:
    - eldönteni mit review-olunk és előkészítjük azt
    - ki vegyen részt? (min 3)
    - ki mit fog csinálni?
        - koordinátor: szerző vagy csapat vezetője, meeting levezénylése, kordában tartása (ne térjen el a tárgytól), nyomon követés
        - lead reader: a vizsgálat vezetője
        - scribe: minden fontosat leír, majd közzétesz
        - user representative: user szószólója
    - meeting meghirdetése, anyagok közzététele

- Meetingen:
    - 15 perc-2óra
    - cél: azonosítás, nem javítás
    - koordinátor vezényli a meetinget
    - lead reader végigvezet mindenkit a review-kon
    - minden hibát feljegyezni
        - ki?
        - hogyan oldja meg?
        - miféle a hiba? (súly, típus)
    - a résztvevők eldöntik, mit kell tenni
        - elfogadjuk
        - elfogadjuk kis változtatás után
        - elfogadjuk sok változtatás után
- Meeting után:
    - scribe közzéteszi 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) + résztvevők és szerepük + célok + akció elemek (bug, deficiencies stb)

- felelős?
- mit csináljunk vele?
- súlyosság és típus

Review szabályok:

- felkészülten érkezz
- szerepednek megfelelően cselekedj
- legyél kooperatív és objektív
- ne késs, ne lógj el!
- megfelelően vegyél részt a meeting utáni feladatokban
- fogadd el a koordinátor és a csapat fennhatóságát

Szoftvertesztelés

- rendszer működésének tesztelése
  - hibák felfedezése
  - megfelel-e a specifikációnak?
- célja:
  - objektumok interakciójának ellenőrzése
  - komponensek jól vannak-e integrálva
  - követelményeket jól teljesíti-e
  - kiadás előtt megtalálni a hibákat
- nem tudja megmondani, hogy jól működik a sw (csak a hibát találjuk meg)
- nehéz
- 30-50% fejlesztési költség
- nem-funkcionális elemek ellenőrzésének egyetlen módja
- statikus verification-nel párban kell használni, hogy a V&V meglegyen

Unit test:

- programozó
- kis végrehajtható elem
  - OO-ban legkisebb a class
  - lehet class, cluster vagy végrehajtható fájl
- magában semmivel se interaktál
- intuitív

Integration test:

- független tesztelő csapat
- egy teljes rendszer vagy alrendszer
- egységek egymástól függnek
- egységek együttműködése
- interfészek az alrendszerben
- Top-down
- Bottom-up

System test:

- független csapat
- teljes integrált alkalmazás
- tulajdonképp amik csak a teljes rendszerben vannak jelen

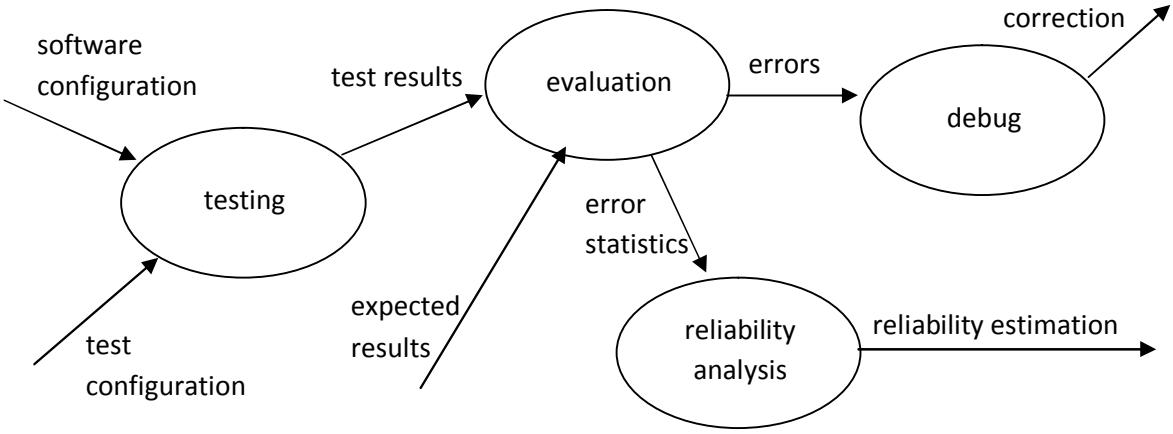
Acceptance test:

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

Hibák típusai:

- error – emberi hiba
- fault (bug) – hiba a programban
- failure – program sikertelensége

Tesztfolyamat:



Minőség dimenziók: FURPS

- Function – elvégzi, biztonságos, elbír-e nagyot
- Usability – szép, értelmes, használható
- Reliability – nem omlik össze, kapcsolatok rendben
- Performance – sebesség, hatékonyság, terhelhetőség elegendő-e
- Supportability – hordozható, installációs problémák stb.

X. CM – Configuration Management

Fő szempontjai:

- Azonosítás
- Menedzselés
- Státus elszámolás & vizsgálat

Fő folyamatai:

- Storage Config Items
- Változás menedzselés
- Felépítés, építkezés menedzselés
- Kiadás menedzselés

CM: szabály bonyolult rendszerek fejlődésének kontrollálására
Szoftver CM: CM szoftverekre specializálva

Azonosítása (Identification)

- config elemek és configok meghatározása:
    - config elem: bármi, ami információt hordoz
    - kommunikáció: e-mail, címlista, IRC log, memo
    - dokumentáció: projekt terv, dokumentációk, user manual
    - implementáció: forráskód, kódkönyvtár, eszközök, futtatható fájlok
- config: config elemek egy csoportja, mely a projekt egy bizonyos állapotát jelöli

Menedzsment

- folyamatok és minőségi elvárások definiálása a projekt megfelelő haladása érdekében
    - config item storage menedzsment:
        - Hol tároljuk az elemeket?
    - változások menedzselése:
        - Ki hagyja jóvá?
        - Ki implementálja őket?
        - Ki ellenőrzi le őket?
    - fejlesztés menedzselése (szoftver gyártás):
        - Miből építkezünk?
        - Mi változott a buildben?
    - kiadvány menedzselés:
        - Mi lesz a kiadvány?
        - Mikor történik a kiadás?

Státus leírás és vizsgálat

- projekt státusának leírása
    - milyen típusú, mennyiségű és súlyosságú hibák vannak észlelve és javítva
    - implementált tulajdonságok
    - teljesített követelmények száma
    - megfelelő mértékegységek használatának biztosítása
- vizsgálat (hol tart a folyamat?)
    - a definiált folyamat szerint haladunk-e
    - próbáljunk 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 készített cuccokkal?
    - Hogyan integráljuk?
    - Hol tároljuk?
- Interfészek:
    - Változtatás menedzsment (Hogyan?)
    - Építés menedzsment (Mi legyen a temp fájlokkal?)
    - Kiadás menedzsment (Hol tároljuk?)

Változtatás menedzsment

- minden változik
    - követelmény
    - hardver kikötés
    - szoftver kikötés
- kontroll: folyamat definiálása
    - változtatást azonosítani, feljegyezni
    - változtatást végigkövetni (állapotgép)
    - a valódi változásnak legyen köze a változtatási kérelemhez
    - Change Control Board
- változtatás a projekt alakulásával szinkronban
    - változtatási kérelmek feldolgozása periodikusan szünetel – 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 feature, új requirement stb.) a hozzá tartozó státus információkkal a projekt egészére kiterjesztve. 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 változtatáshoz).

Felépítés menedzsment

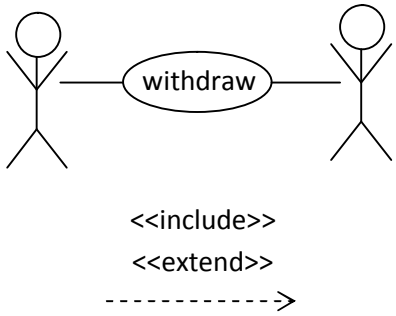
- baseline: pillanatfelvétel egy bizonyos időben minden egyes termék egy bizonyos verifikációjáról a projektben
- delta: egy elő baseline-tól a következőig történt változások
- Minek?
    - megismételhető
    - követhető
    - jelenthető

Kiadás menedzsment

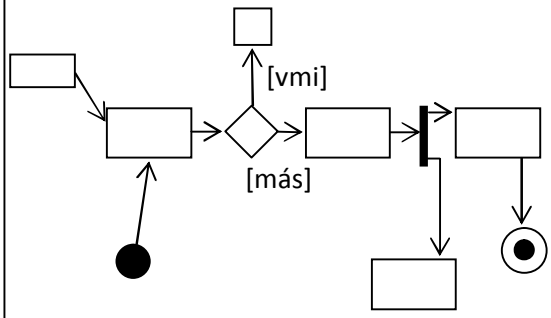
- definíciók:
    - verzió: funkcionálisan eltérő változat (egy rendszernek)
    - variáns: funkcionálisan egyezik, nem-funkcionálisan eltér
    - kiadás: olyan verzió, amit a developer team-en kívül is publikálnak
    - javított kiadás: verzió + variáns (de nem új release!)
    - check-out: adott elemet szerkesztésre visszatartani
    - check-in: változtatott elemet publikálni
        - változtat, update-el, merge-el
- verzió név probléma: Build 1234 vagy Ultimate Edition XP SP2?
- verziókat számozzuk
    - tulajdonság-orientáltan
    - változtatás-orientáltan

Behavioral Models

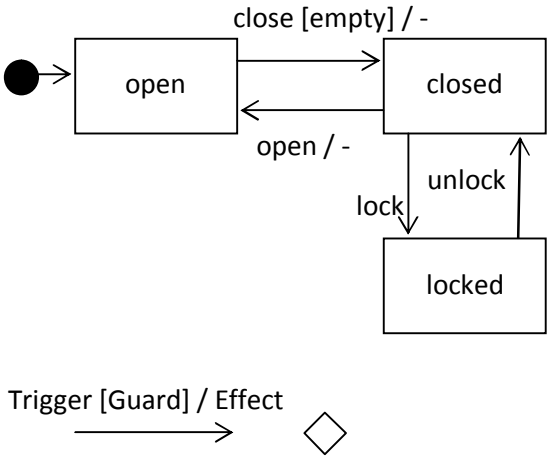
ud – Use Case



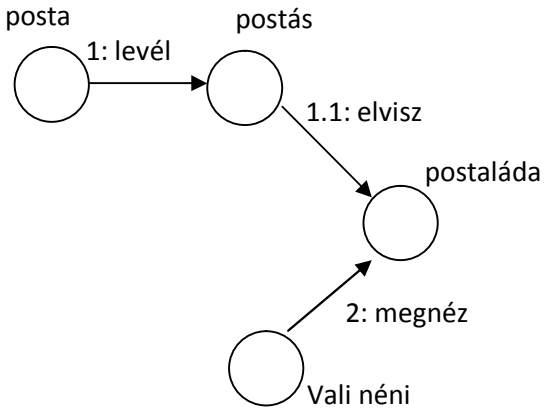
ad – Activity



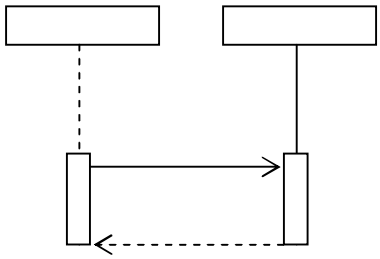
sm – State Machine



cd – Communication

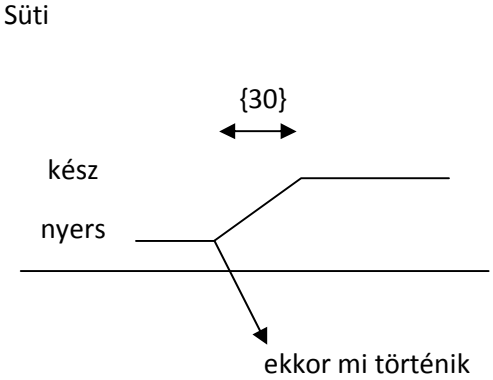


sd – Sequence

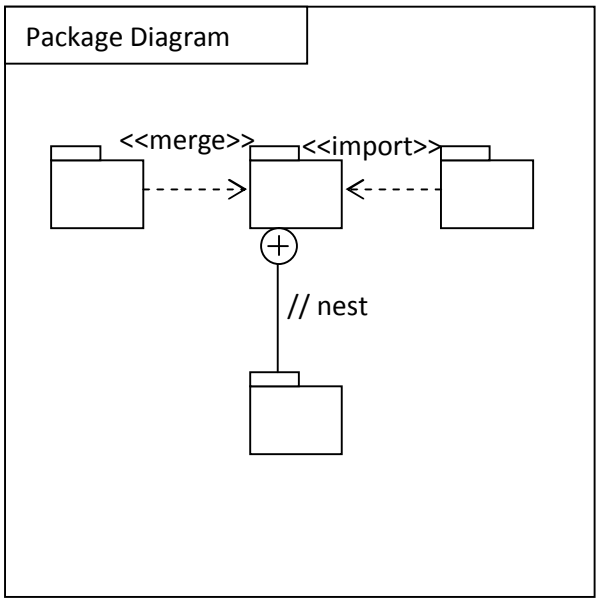


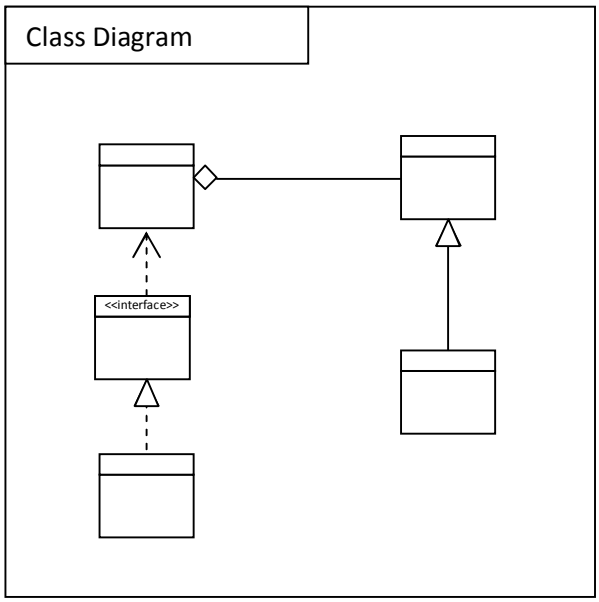
válaszüzenet
aszinkron
szinkron

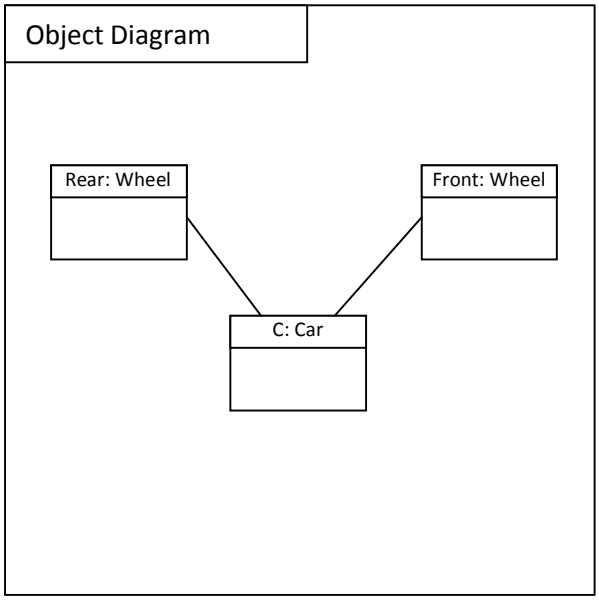
td – Timing

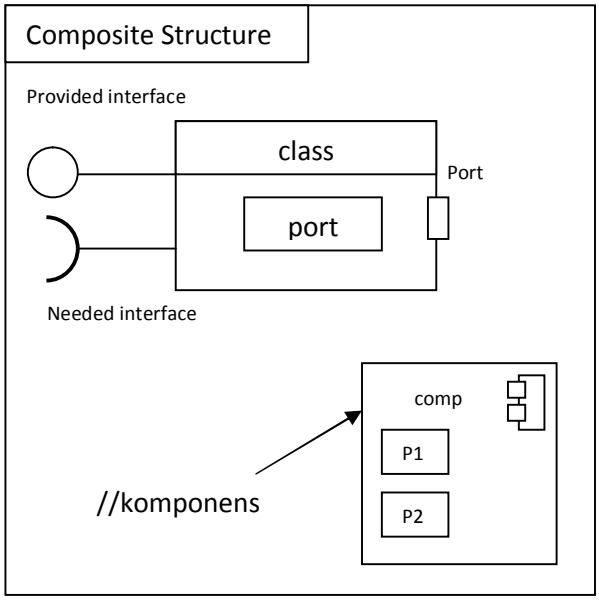


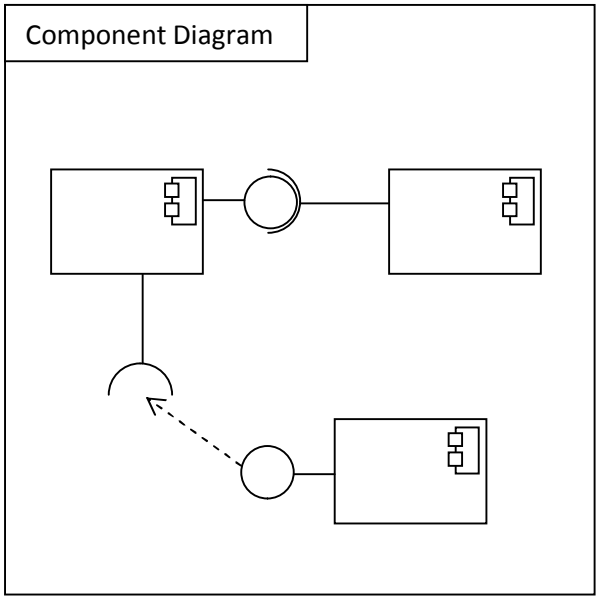
UML 2

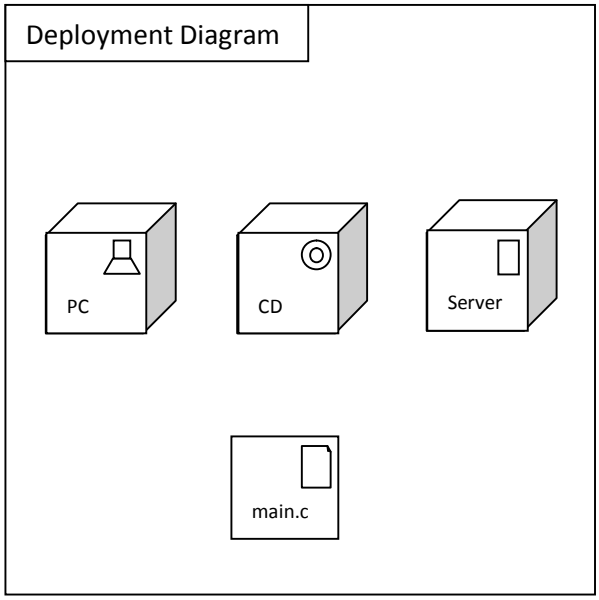

Package Diagram
<<merge>>
<<import>>
// nest


Class Diagram
<<interface>>


Object Diagram
Rear: Wheel
Front: Wheel
C: Car


Composite Structure
Provided interface
class
Port
port
Needed interface
comp
P1
P2
//komponens


Component Diagram


Deployment Diagram
PC
CD
Server
main.c

Tartalom