Defininiálja a technológia (engineering, mérnökség) fogalmát öt komponensének megadásával!

Gyakorlati problémák megoldására szolgáló **dolgok,** a **társadalom** által szabályozott **tudományos** alapokon álló **gazdaságos** előállításának miként je.

Rajzolja fel a tesztelés információs folyamatának adatfolyamábráját!

Sorolja fel a JSD 6 lépését!

1. Entitás akció (entity action)
2, Entitás élettörténet (entity structure)
3. Kiinduló modell (initial model)
4. Funkció (function)
5. Időzítés (timing)
6. Megvalósítás (implementation)

Nevezze meg sorrendben a CMM (Capability Maturity Model) szintjeit!

1. Kezdetleges (initial)
2. Ismétlődő (repeatable)
3. Definiált (defined)
4. Irányított (managed)
5. Optimalizált (optimizing)

Mi a CMM negyedik szintjének neve, jellemezze röviden!

Menedzselt, irányított (managed):

Mind a szoftver fejlesztési folyamat, mind a termék számszerűen mérhető és kontrollálható.
Both the **software process** and **products** are **quantitatively understood and controlled**.

Nevezze meg a funkciópont elemzés (Funkction Point Analiszs, FPA) komponenseit!

External Inputs
External Enquiry
External Interface Files
External Outputs
Internal Logical Files

Mi a különbség a funkciópont elemzésben (FPA) alkalmazott External Output (EO) és External Inquiry (EQ) között?

	EO	EQ
számított eredményeket ad	igen	nem
módosítja valamely ILF-et	igen	nem

Az UML komponensek fajtái

deployment (a software indításához kell, pl class)
execution (a végrehajtás közben keletkezik, pl file)
work product (a fejlesztő munka terméke, pl *.h)

A minőség az alábbi nézőpontokból vizsgálható (views of quality)

- Transzcendentális (transzcendental)
- Felhasználó (user)
- Gyártás (manufacturing)
- Termék (product)
- A pénzért kapható érték (value-for-money)

A minőség nézőpontjai közül a szoftver technológia melyiket tekinti alapvetőnek?

- Gyártás (manufacturing)

Az UML szerint az operációk konkurrenciájának szemantikája lehet:

- Szekvenciális (sequential)
- Őrzött (guarded)
- Konkurrens (concurrent)

A "végrehajtási szemantika" (execution semantics) mely OO alapfogalmakhoz társítható?

- Az operációhoz

Az OO alapmodell milyen végrehajtási szemantikákat definiál és azokat hol és hogyan jelöli?

at-most-once	-
best effort	oneway a szignatúrában

Írja fel egy operáció szignatúrájának általános formáját!

- [oneway] <op_type_spec> <identifier> (param1,..., paramL) [raises(except1, ..., exceptN)] [context(name1,..., nameM)]

Egy modul egy másik modul használata során record (vagy struct) paramétert ad át. Milyen a két modul közötti csatolásban (coupling) a "kommunikáció fajtája" (kind of communication)?

- Stamp (bélyeg)

Egy A objektum miközben a B objektum egy metódusát hívja, paraméterül egy C objektumot ad át. Milyen a hívó és hívott objektumok (A és B) közötti csatolásban (coupling) a "kommunikáció fajtája" (kind of communication)?

- Stamp (bélyeg)

Adja meg a szoftver munka kiszámításának képletét, a képletben szereplő értékek jelentését, valamint a képletből levonható legfontosabb következtetést!

- $E=V/1=V^2/V^*$
- V= tényleges programtérfogat
- V^*= elméleti program
- I= implementációs szint
- A programot részekre kell bontanti!

Adja meg a felsorolt komponenst rendszerekben alkalmazott jellemző adaptációs módszereket!

Rendszer	*Adaptáció*
Modul	Tipikus kódolás, de nyelvfüggő
Keretrendszer	Öröklés, delegálás
CORBA és társai	Csomagoló osztályok
Architektúra rendszerekben	Konnektorok
Aspect oriented programming	Szövés

Mit nevezünk "temporális kohézió"-nak és mikor használjuk?

- A kohézió temporális, ha a vizsgált egységben (modulban, objektumban) szereplő elemek közötti kapcsolat lényege nem a funkcionalitás, hanem az egyidejűség. Olyankor használjuk, amikor a különböző tevékenységek időbelisége nagyon kötött, így inicializálásnál, kivételes helyzetek kezelésénél, a programok (taskok) futásának lezárásánál, felfüggesztésénél.

Milyen mennyiségek között definiál kapcsolatokat a CCM (Constructive Cost Model?

- MM – erőfeszítés emberhónap
- KS – kilosource
- T – fejlesztési idő

A Petri-hálót egy 4 elemű algebrai struktúra (P, T, A, M) írja le. Definiálja (algebrailag és szövegesen) az egyes elemeket!

P={P1, ...Pn} – place-ek halmaza
T={T1, ...Tm} – transition-ök halmaza
$A \subseteq \{(P \times T) \cup (T \times P)\}$ – élek halmaza
$M = P \rightarrow N$ - markerezés

Kohéziós osztályok

funkcionális

Ez az ideális. A modul egyetlen jól definiált célt szolgál.
Példa: sqrt(); komplex számos osztály

szekvenciális

Ilyenkor azért vannak egy függvénybe csoportosítva a műveletek, mert az egyis művelet outputja a következő inputja.
Példa: getValidRecord(); számbevitelnél, ha billentyűről olvas a modul, utána ellenőrizheti, a beolvasott string tényleg szám-e

kommunikációs

A modulhoz tartozó műveletek mind hasonló adatokon végeznek műveleteket, de más közük nincs.
Példa: Egy könyvtári nyilvántartó programban az a modul amely ellenőrzi, hogy bent van-e a könyv, ha nincsen, akkor mikor vették ki, és hogy a könyv ISBN száma létező, korrekt adat-e, nem kamu. Ideális esetben ezeket nem kellene egy függvénybe csoportosítani, hanem mindháromra külön függvény kellene. Azért szokott előfordulni, mert csábító a „ha már egyszer úgyis lekértük ezt a rekordot, akkor még ezt igazán megcsinálhatjuk” mentalitás.

procedurális

A modul által végrehajtott műveletek logikailag egymás után következnek. Kicsit hasonlít a szekvenciálishoz, de az elvégzett műveletek nem annyira szorosan kapcsolódnak, mert egy művelet outputja nem feltétlenül kell, hogy a soron következő művelet inputja legyen.
Példa: printf(); write(); Egy nyomtatásért felelős függvény a PostScript küldése előtt ellenőrzi, hogy be van-e kapcsolva a nyomtató, ha nincsen, ezt megteszi és takarékossági okokból fekete-fehér tintára állítja a nyomtatót. Kényelmes tud lenni, de az újra felhasználhatóságnak keresztbe tesz, mert ha később fájlba szeretnénk nyomtatni, akkor már sajnálni fogjuk hogy a modul össze-vissza cseveg a hardverrel. Mellesleg ezért van az hogy JAVA-ban nem egyetlen fopen()-nel szoktunk fájl-t nyitni. Gyakran kényelmesebb egyetlen függvény, de ez nem újrahasználható.

temporális

Időben egyszerre elvégzendő műveletek kerülnek egy modulba.
Példa: init(), amely inicializálja a fájlkezelést és az üzenetbuffert; shutdown(), ami felszabadítja a lefoglalt; Egy errorHandler függvény ami bezárja a nyitott fájl-okat, értesíti a felhasználót a hibáról, és csinál egy logfile-t. Abban tér el a procedurálistól, hogy az elvégezett műveletek nem lépcsőfokai egy jól definiált cél elérésének, hanem függetlenek egymástól, csak ugyanakkor kell őket megcsinálni. memóriát és lezárja a fájlokat

logikai

Egy modul által elvégzendő műveletek között csak laza logikai szál van.
Példa: 12 function-t neve szerint abc-be rendezünk, majd három modulra vágunk szét

esetleges

Ilyenkor teljesen logika mentes hogy mit képes elvégezni a modul.
Példa: Egy webprogramozáshoz használt Utility osztály, ami dekódolja a http GET üzeneteket, konvertál iso-8859-2 és unicode között, és le tudja ellenőrizni egy linkről hogy halott-e.

A három rétegű szoftver architektúrában milyen logikai rétegeket definiáltunk?

GUI – felhasználói felület
BOM – alkalmazási modell
DB – adatbázis

A C osztály M metódusából a Demeter törvény szerint az alábbi program-elemek érhetők el:

Az M paraméterei
this, super
A C osztály és példány változói (attribútumok)
globális változók
Az M-ben létrehozott átmeneti változók
az M által létrehozott objektumok

Ha a hagyományos MODULÁRIS programokból építünk komponens rendszert, akkor

mi a kommunikáció módja: eljárás hívás
mik az interfészek: paraméteres eljárások, globális változók
létezik-e genericitás: esetleges (Ada)
hogyan adaptálunk: a kód átírásával

Ha a hagyományos OO programokból építünk komponens rendszert, akkor

mi a kommunikáció módja: polimorf eljárás hívás
mik az interfészek: polimorf metódusok, osztály és objektum változók
létezik-e genericitás: gyakran előfordul (C++)
hogyan adaptálunk: öröklés, delegálás

Az Aspektus Orientált Programozásban

mi biztosítja a genericitást: az aspektusok szövése
mik az interfészek: csatlakozó pontok (join points) a szöveshez
hogyan adaptálunk: ragaszkodó kódok és csomagolók szövése, aspektusok változtatása

Milyen kockázatkezelési stratégiákat alkalmazunk a tervezésben?

A kockázat valószínűségének csökkentését
A bekövetkező káros hatások minimalizálását
A nem kívánatos eseményt követően felmerülő tevékenységek tervezését

Melyek a kockázatkezelési folyamatai (risk management process)?

Kockázatok azonosítása (identification)
Kockázatok elemzése (analysis)
Kockázatok tervezése (planning)
Kockázat felügyelet (monitoring)

A kockázatok tervezése során (risk planning) milyen stratégiákat alkalmaznak?

Elkerülés (avoidance)
Minimalizálás (minimisation)
Folytatás (contingency)

Nevezze meg és jellemezze 1-1 mondattal az életciklus modelljének fázisait!

Inception phase Kezdeti fázis):
Az üzleti alapok megtervezése, a projekt tevékenységi körének meghatározása. A projekt részletezése.
Establishing the business rationale for the project, and decide on the scope of the project. Specifying the project vision.

Elaboration phase (Tervezési fázis):
A szükséges tevékenységek megtervezése, az erőforrások valamint a sajátosságok meghatározása, az achitektúra kivitelezése.
Planning the necessary activities and required resources; specifying the features and designing the architecture.

Construction phase (Kivitelezési fázis):
A működő rendszer kiépítése, tesztelése a korábbi kidolgozási fázis tervei alapján.
Building and testing a working system according to the previously elaborated plans as a series of incremental iterations.

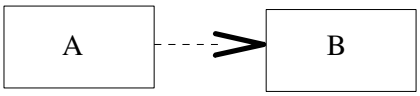
Transition phase (Átmeneti fázis):
A termék a felhasználók számára való biztosítása (gyártás, szállítás, betanítás).
Supplying the product to the user community (manufacturing, delivering, and training).

A RUP a fogalmi modell megalkotásával kapcsolatosan a “térképész-elv” követését javasolja. Mit mond ki a “térképész-elv”?

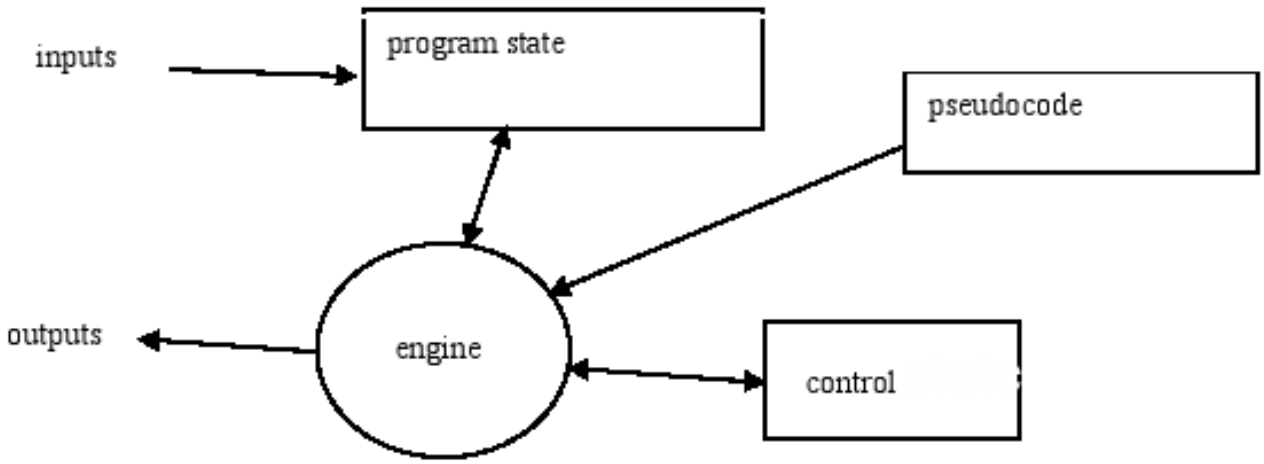
A valóságban létező elnevezéseket használd!
Ne foglalkozz a lényegtelen, jelentéktelen dolgokkal!
Ne foglalkozz a valóságban nem létező dolgokkal!

Mit jelent az, hogy A függ B-től (A depends on B) és hogyan jelöli az UML?

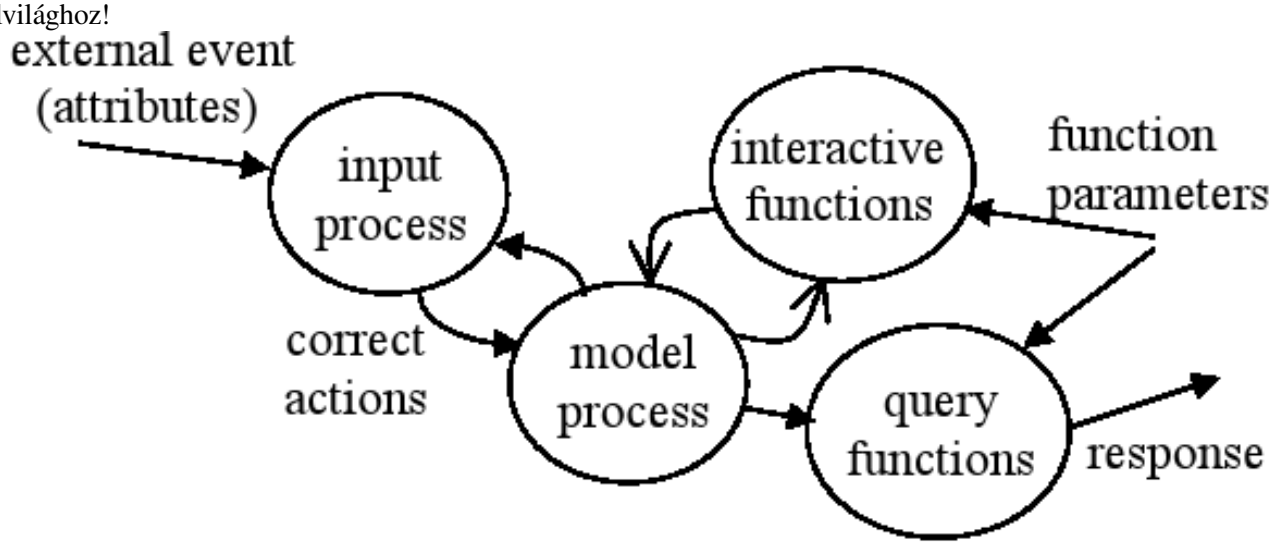
Reláció. Ha B specifikációjának változása A megváltozását vonja maga után.



Rajzolja fel az interpreter szoftver architektúra mintát!
Rajzolja fel az interpreter szoftver architektúra vázlatát!



Rajzolja fel DFD ábrán, hogy a JSD-ben a modell és funkció processzek hogyan kapcsolódnak egymáshoz és a külvilághoz!



UML relációk és elemek

implementáció/realizáció: interface függvényeit megkapja ------▷
tartalmazás (aggregation): gyenge megkötés, A B-n kívül másból is áll ——◇
kompozíció: erős megkötés, A csak B-ből áll ——◆
függőség (dependency): - - - - >
osztálymetódus: statikus, aláhúzott
öröklés (inheritance): ——▷
asszociáció (association): A ismeri B-t, A ——> B
absztrakt: lesármazottak megkapják az abstrajk függvényt, *dőlt*
sztereótípus: ős
szubtípus: leszármazott
+ public, - private, # protected

Milyen minőségi jellemzőket tesztelünk a különböző (FURPS) típusú tesztekkel?

Funkcionalitás (functionality)
Használhatóság (usability)
Megbízhatóság (reliability)
Teljesítőképesség (performance)
Támogatottság (supportability)

Adja meg a fejlesztési folyamatban előforduló tesztelési fokozatokat (stages)!
A “V model” szoftver életciklus alapján milyen tesztelési szinteket azonosíthatunk?

Egység/modul teszt (unit test)
Integrációs teszt (intergration test)
Rendszer teszt (system test)
Átadás/elfogadás teszt (acceptance test)

Melyek a fő típusai a kohézív osztályokat összefogó domain-eknek?

Foundation (Abstract/Math Unit Struct)
Business (Role Relationship)
Architectural (HCI DB Com)
Application

Mit jelentenek az alábbi fogalmak?

At-most-once:

Ha a művelet sikeresen tér vissza, pontosan egyszer ment végbe; ha kivételt ad vissza, at-most-once, vagyis legfeljebb egyszer ment végbe. A kliens mind szinkron, mind halasztott szinkron módon is végrehajthat at-most-once műveletet.
If an operation request returns successfully, it was performed exactly once; if it returns an exception indication, it was performed at-most-once. A client is able to invoke an at-most-once operation in a synchronous or deferred-synchronous manner

Best-effort:

Best-effort csak kérésre hívódik meg. Nem garantálja a művelet sikerét, de mindent megtesz azért, hogy az legyen.
Best-effort is a request-only operation (i.e., it cannot return any results and the requester never synchronizes with the completion, if any, of the request)

Minek a végrehajtásával kapcsolatosan definiáltuk őket: Operation

Melyik az a három elv, amelyet mind adatok, mind utasítások komponálásakor alkalmazunk?

Szekvencia
Iteráció
Szelekció

Definiálja a szoftver hibával kapcsolatos alábbi fogalmakat:

Error:

A hiba emberi cselekedet, vagy emberi mulasztás következtében keletkezik.
Human action or omission that results in a fault.

Fault:

A fault egy szoftver hiba (nem megfelelő lépés, folyamat, vagy adat meghatározás).
Fault is a software defect (incorrect step, process or data definition) that causes a failure.

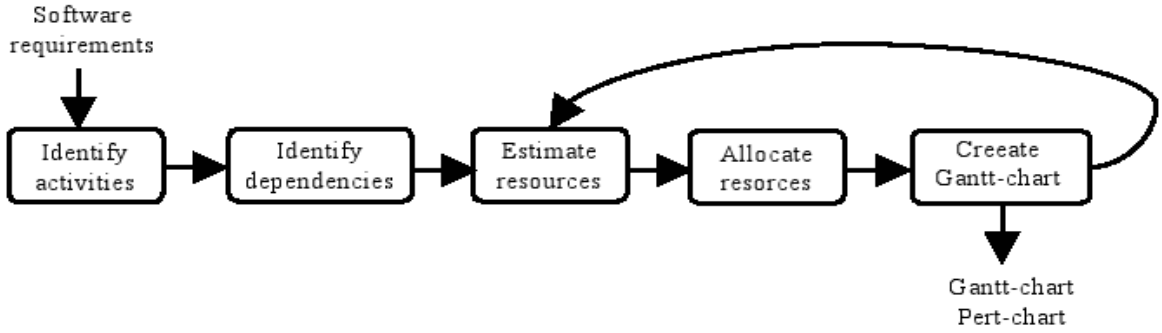
Bug:

Megegyezik a Fault-tal.
Same as Fault.

Failure:

Amikor a szoftver nem képes teljesíteni a szükséges feltételeket (pl. túl lassú).
The inability of a software to perform its required functions within specified performance requirements.

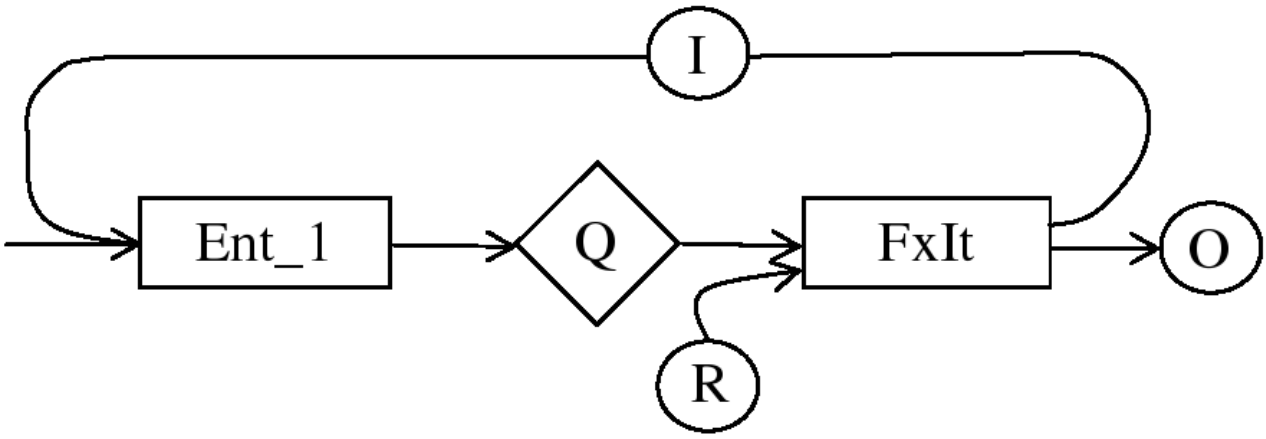
Hogyan származtatja a szoftver követelményekből az ütemezéssel kapcsolatos Gantt és Pert diagramokat? Rajzolja fel a folyamat adatfolyamábráját!



Mik a csatolás dimenziói?

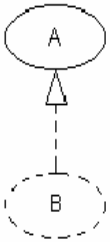
Kommunikáció típusa (kind of communication)
Kapcsolat mérete (size of connection)
Kapcsolat ideje (time of connection)

A JSD-ben az Ent_1 modell processzhez kapcsoljon a modell állapotán alapuló, felhasználói kérésre induló FxIt iteratív funkciót! Rajzoljon SSD-t!



Hogyan értelmezzük az UML-ben a szekvenciális konkurrenciát (a konkurrencia szemantikája szekvenciális)?

Az objektum hívói meg kell, hogy szervezzék maguk köztük azt, hogy egyidejűleg csak egyikük használhatja.
Callers must coordinate outside the object so that only one flow is in the object at a time.



Mi A és B az alábbi UML diagramon?

A: use-case
B: kollaboráció

Mi(k)hez kapcsolódik a CORBA rendszerben az "IDL csontváz" (IDL Skeleton)?

Objektum adapter
Szervant

Mi(k)hez kapcsolódik a CORBA rendszerben az "IDL betét" (IDL Stub)?

ORB mag
Kliens

Mi a Lack of Cohesion Metric (LCOM) képlete?

LCOM=|P|-|Q|

Mit nevezünk "legacy software"-nek? Miért fontos a szoftver technológia szempontjából?

1. Azon szoftver, mely gazdasági élete bár letelt, továbbra is tulajdon.
1. Has exceeded its economic life time - but cannot be abandoned.
2. Nagy karbantartási igény, refactoring, re-engineering.

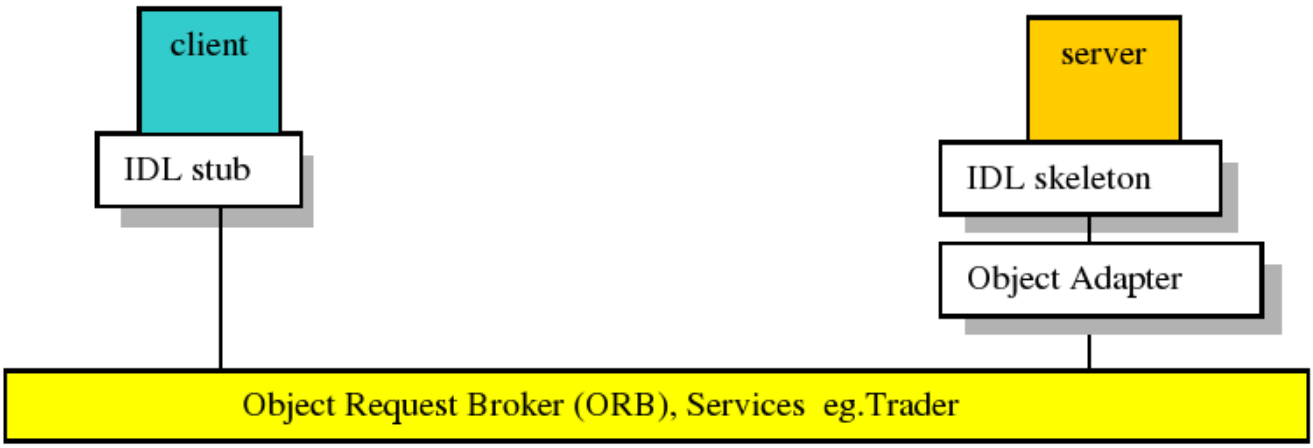
Milyen integrációs (vagy tesztelési) stratégia esetében használunk teszt ágyakat (test bed)? Mi a funkciója a teszt ágynak?

1. Bottom-up
2. Környezet, amelyben az integrált unitok együttes működése tesztelhető.

Mit jelent a "szoftver konfigurációs menedzsment" (CM)? Mi a "konfiguráció"?

1. A CM szabja meg a bonyolult rendszerek fejlődési módját; szoftver CM ennek a specializációja számítógépes programokra, dokumentumokra.
1. CM is the discipline of controlling the evolution of complex systems; software CM is its specialization for computer programs and associated documents.
2. A konfiguráció különböző konfigurációk összessége, melyek kijelölik a projekt állapotát.
2. A configuration is a selection of configuration items designating a state of the project.

Rajzolja fel a CORBA architektúrát!



A tervezési elvek között bevezettük a kohézió és csatolás fogalmait. Mit jelentenek? Milyen fő típusaik vannak?

Kohézió: Mennyire kell ismerni kívülről egy (al)rendszer belső szerkezetét. Mennyire fekete a doboz?
- *Típusai*: funkcionális, szekvenciális, kommunikációs, procedurális, temporális, logikai, véletlen

Csatolás: Egy (al)rendszer komponensei közötti kapcsolat erőssége.
- *Dimenziók*:
  - *kapcsolat módja*: paraméter, adat, vezérlés, tartalmi
  - *kapcsolat ideje*: run-time, load, link, compile, source
  - *kapcsoló adatok menyisége*: kevés, sok

Sorolja fel a komponensek közötti kommunikáció módjait!

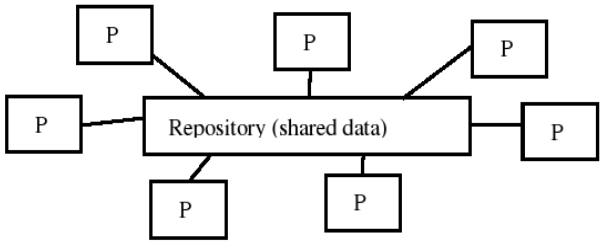
Eljárás hívás
Visszahívás (callback)
Távoli eljáráshívás (remote procedure call, rpc)
Közös memória alkalmazása
FIFO, pipe, socket
Adattáblák, struktúrált adattáblák (blackboards, structured blackboards)
Tranzakciós szolgáltatások (pl. DBMS)
Kód migráció (pl. Applet)

Jellemezze a Blackboard (tábla) szoftver architektúrát! Adja meg a komponenseket, az architektúra alkalmazásának előnyeit és hátrányait!

Komponensek:
- Szeparált, független programok, alrendszerek, amelyek közös memórián keresztül működnek együtt
- Központi adattár (adatbázis)

Előnyök:
- Az adatok, a vezérlés és a feldolgozás jól elkülönül
- Kisebb adathibák tűrése és korrekciója

Hátrányok:
- Bonyolult tesztelni
- Alacsony hatékonyság
- Nagy fejlesztési erőfeszítést igényel



Adja meg a Rational Unified Process három fő jellemző tulajdonságát!

Use-case vezérelt
Architektúra központú
Iteratív és inkrementális

Mi a döntés hasítás?

Vezérlési hatáskör < Döntési hatáskör

Mit jelent az objektum változókon bevezetett **tipizálás** és **kötés** fogalma? Milyen összefüggés van közöttük?

Tipizálás: Milyen objektumok tehetők a változóba (statikus – fordításkor; dinamikus – futáskor definit)

Kötés: Kihez kapcsolódnak a műveletek (statikus – a változóhoz; dinamikus – az értékhez)

Kapcsolat:

	statikus tipizálás	dinamikus tipizálás
statikus kötés	nem OO	értelmetlen
dinamikus kötés	normál OO (Java, C++)	Smalltalk, Excel cella

Adja meg a program hosszára vonatkozó Halstead-féle mér\u0151számot és a programtérfogat definícióját!

$\xi 1$= különböző operátorok száma
$\xi 2$= különböző operandusok száma
$\xi = \xi 1 + \xi 2$ szótár
N1= összes operátor száma
N2= összes operandus száma
N=N1+N2 program hossza
$N = \lambda o \gamma 2 \xi 1 * + \xi 2 * \lambda o \gamma 2 \xi 2$
$\varsigma = N * \lambda o \gamma 2 \xi$

A szoftver projekt COCOMO szerint költségbecslése során alkalmazott egyik módosító/hangoló tényező (Effort Adjusment Factor) a “turnaround time” (TURN). Magyarázza el a fogalom jelentését!

Egy hiba javítását követően a program új változatának előállításához (fordítás, szerkesztés, konszolidálás, stb) szükséges idő.

Konfiguráció menedzsment (CM)

Key Aspects:
- Identification
- Management
- Status Accounting & Auditing

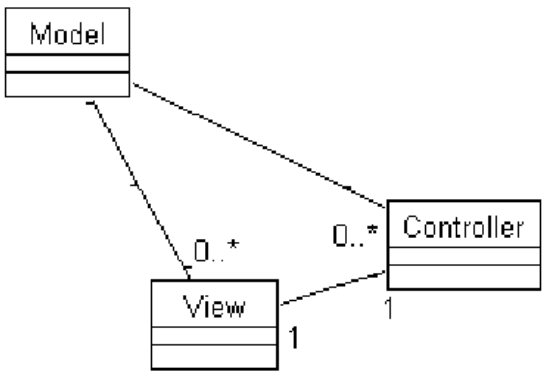
Key Processes:
- Storage Configuration Items
- Change Management
- Build Management
- Release Management

CM-Tools:
- Version Control
- Change Control
- Build Management
- Communications

RUP Workflow Test részei:

Requirements (Use-Case model)
Analysis model
Design model
Implementation model
Deployment model

Ismertesse az MVC modelt! Rajzolja le a classdiagramot és definiálja az osztályok felelősségeit!



M (model): a business logic
V (view): megjelenés
C (controller): a V és M vezérlése

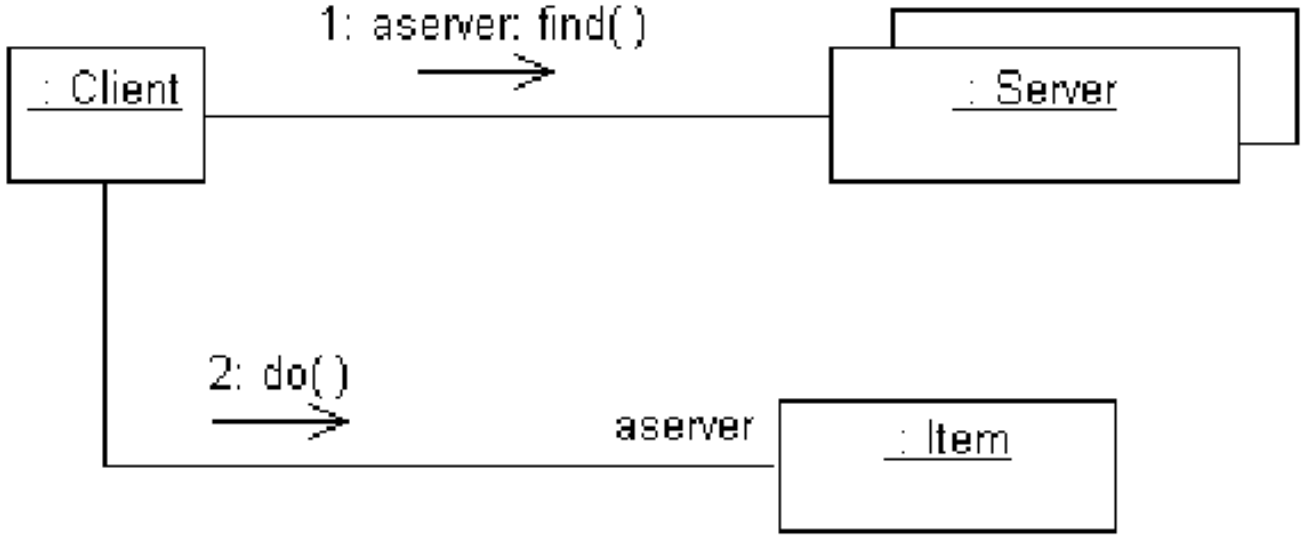
Mi a különbség a walkthrough és az inspection között?

walkthrough	*inspection*
mindig alkalmazható	definit szintaxis
beszámoló	előzetes anyag

Nevezze meg a **CASE rendszerek** fontosabb komponenseit!

- Repository
- Project management support
- Structured diagramming tools
- Document generation facilities
- Skeleton code generator
- Query language facilities
- Form creation tools
- Design analysis and checking tools
- Export/import facilities

Mi a multiobject? Hogyan jelöljük és milyen diagramon fordul elő?



Mik a karbantartás költségeit befolyásoló technikai tényezők?

- Objektumok, modulok függetlensége
- Programozási nyelv
- Programozási stílus
- Verifikáció és validáció minősége
- Dokumentáció minősége

Mi a CASE rendszerek “export/import facilities” alrendszerének feladata?

- A modellnek vagy egyes részeinek a mozgatása különböző modellek/CASE rendszerek között.
- Kulcskérdése az export-import formátum (egységessége).

Melyek a hagyományos komponens rendszerek?

- UNIX
- XML
- Modul
- Osztályok és öröklés
- Keretrendszerek

A tesztelés kapcsán az egyik legfontosabb kérdés, hogy “mikor fejezzük be a tesztelést?”. Nyilván akkor, amikor a programbn maradó hibák száma az előírt érték alá csökken. A maradó hibák számát azonban csak becsülni tudjuk. Milyen módszerrel?

- Vadszámlálás