1. Mi a szoftver, milyen részekből áll és milyen típusait különböztetjük meg? Mik a szoftverfejlesztés általános lépései?
Számítógépes programok és a hozzá kapcsolódó dokumentációk (pl. követelmények, tervezési modellek és felhasználói kézikönyvek).
Típusai:
• **Általá**nos: felhasználók széles rétege számára fejlesztett és általuk használt szoftver.
• **Egyedi**: egy megrendelő egyedi igényei szerint készült.
A szoftverfejlesztés általános lépései:
• **Specifikáció**: mit kell a rendszernek tudnia és mik a fejlesztési kényszerek, kötöttségek.
• **Fejlesztés**: a szoftver rendszer megalkotása.
• **Validáció**: ellenőrzés: a szoftver azt csinálja, amit a megrendelő akar?
• **Evolúció**: A szoftver változó igények szerinti továbbfejlesztése.

2. Mik a szoftvergyártás általános modelljei?
• **Vízesés modell**.
• **Iteratív fejlesztés**.
• **Komponens alapú fejlesztés**.

3. Hogyan alakul a rendszerfejlesztés költségeinek eloszlása különféle modellek alkalmazása esetén, egyedi szoftverek fejlesztése során?
Vízesés modell:
Specification: 15, Design: 25, Development: 20, Integration & Testing: 40
Iteratív fejlesztés:
Specification: 10, Iterative Development: 60, System Testing: 30
Komponens alapú fejlesztés:
Specification: 20; Development: 30; Integration & Testing:50;
Egyedi szoftver:
System Development: 100, System evolution: 300

4. Hogyan alakul a rendszerfejlesztés költségeinek eloszlása általános szoftverek fejlesztése során?

Specification: 5, Development: 35, System Testing: 60;

5. Mik a szoftverfejlesztési módszertanok, mik ezek legfőbb elemei?
Olyan strukturált szoftverfejlesztési módszerek, amelyek tartalmaznak rendszermodellező eszközöket, jelölési konvenciót, szabályokat és tervezési ajánlásokat, valamint fejlesztési útmutatót.
Elemei:
• **Modell leírások**: a létrehozandó grafikus modellek leírása.
• **Szabályok**: a rendszermodellekre vonatkozó kényszerek.
• **Ajánlások**: a helyes tervezési megoldásokra vonatkozó tanácsok.
• **Fejlesztési útmutató**: a modellfejlesztés során végrehajtandó tevékenységek sorozata.

6. Mi az a CASE?
Olyan szoftver rendszerek, amelyek a szoftverfejlesztési folyamatot automatikus eszközökkel támogatják.
• Upper-CASE: a fejlesztés korai fázisait támogató eszközök.
• Lower-CASE: a fejlesztés későbbi fázisait támogató eszközök.

7. Sorolja fel a jó szoftver 5 ismérvét!
A felhasználó által megkívánt **funkcionalitást és teljesítményt** szolgáltatja, **jól karbantartható**, **megbízható**, **hatékony** és **befogadható**.

8. Mik a szoftverkészítés legfőbb kihívásai korunkban?
• **Heterogenitás**: szoftverkészítést heterogén platformokra és végrehajtási környezetekre.
• **Határidők**: gyorsabb fejlesztés és átadás.
• **Bizalom**: felhasználók bizalmát megnyerni képes fejlesztési technológia.

9. Sorolja fel a szakmai felelősség 4 alapvető problémáját!
• **Titoktartás**
• **Felkészültség**
• **Szellemi tulajdonok**
• **Technikai visszaélés**

10. Sorolja fel az IEEE/ACM etikai kódex 8 alapelvét és magyarázza el ezek jelentését!
• **Közérdek**: a szoftvermérnököknek a köz érdekének megfelelően kell cselekedniük.
• **Ügyfél és alkalmazó**: a szoftvermérnöknek a megrendelő és az alkalmazó érdekében kell eljárnia, a közérdek figyelembevételével.
• **Termék**: a szoftvermérnöknek biztosítania kell, hogy termékei a lehető legmagasabb szakmai színvonalat érjék el.
• **Ítélőképesség**: a szoftvermérnökök szakmai ítéleteit önállóan és függetlenül kell meghoznia.
• **Menedzsment**: a menedzserek és egyéb vezetők kötelessége az etikus szoftverfejlesztés és -karbantartás biztosítása.
• **Szakma**: a szoftvermérnöknek a szakma jó hírét a köz érdekével öregbítenie kell.
• **Munkatársak**: a szoftvermérnöknek támogatnia kell munkatársait.
• **Önfejlesztés**: a szoftvermérnöknek folyamatosan fejlesztenie kell szakmai tudását.

11. Mit értünk rendszer alatt?
Kapcsolódó komponensek halmaza, amelyek egy közös cél érdekében működnek együtt.
A rendszer tartalmazhat szoftvert, mechanikus és elektronikus hardvert.
A rendszert emberek is üzemeltethetik.

12. Mik a technikai rendszerek és az ember-gép rendszerek alapvető tulajdonságai?
Technikai rendszerek: hardvert és szoftvert tartalmazó rendszerek. Az operátorokat és a rendszert működtető eljárásokat nem tekintjük a rendszer részének.
Ember-gép rendszerek: olyan rendszerek, amelyek technikai rendszereket is tartalmaznak, csakúgy, mint a rendszert működtető eljárásokat és a technikai rendszerrel kapcsolatot tartó embereket.
Alapvető tulajdonságok:
• **Globális/eredő tulajdonságok**: a rendszerkomponensektől és azok kapcsolatától függenek.
• **Nem determinisztikus viselkedés**: azonos bemenőjelre nem mindig azonos kimenőjelet produkálnak.
• **Szervezeti céloktól való komplex függés**: nem csak a rendszertől függ, hogy az mennyire képes a szervezet céljait szolgálni.

13. Mik az eredő tulajdonságok, mik ezek ismérvei? Soroljon föl példákat.
Az egész rendszerre vonatkozó tulajdonságok, melyek nem származtathatók a komponensek tulajdonságaiból.
• A globális (eredő) a rendszerkomponensek kapcsolatából adódnak.
• Ezen jellemzők csak akkor mérhetők, ha a komponensek rendszerré történő integrációja megtörtént.
Pl.:
Térfogat: a rendszer teljes térfogata a komponensek összeállításának mikéntjétől függ.
Megbízhatóság: a rendszer megbízhatósága függ a komponensek megbízhatóságától.
Javíthatóság: jellemzi, hogy milyen egyszerű a rendszert javítani, miután a hibát észlelték.
Használhatóság: milyen könnyű a rendszert használni.

14. Milyen két alapvető eredő tulajdonság-típust ismerünk? Adjon rájuk példákat is.
Funkcionális tulajdonságok: akkor láthatók, ha a rendszer valamennyi eleme egy cél elérése érdekében közösen dolgozik.
Pl.: egy kerékpárnak akkor funkcionális tulajdonsága, hogy közlekedési eszköz, ha azt alkatrészeiből már összeszerelték.
Nem-funkcionális tulajdonságok: ezek a rendszernek a környezetével való kapcsolatát jellemzik. Számítógépes rendszereknél gyakran kritikus tulajdonságok: amennyiben egy minimális szintet nem érik el, a rendszer könnyen instabillá válhat.
Pl.: megbízhatóság, teljesítmény, biztonság.

15. Mi befolyásolja a megbízhatóságot?
• **Hardver megbízhatóság**: mennyi a hardver komponens meghibásodási valószínűsége és mennyi ideig tart ennek a komponensnek a javítása?
• **Szoftver megbízhatóság**: mekkora annak valószínűsége, hogy egy szoftver komponens hibás eredményt produkál?
• **Operátor megbízhatósága**: mennyire valószínű, hogy a rendszeroperátor hibázik?

16. Sorolja fel a rendszerkövetelmények 3 típusát!
• **Absztrakt funkcionális követelmények**: a rendszer funkcióit absztrakt módon definiáljuk.
• **Rendszertulajdonságok**: az egész rendszerre vonatkozó nem funkcionális követelményeket definiáljuk.
• **Nem kívánatos tulajdonságok**: nem megengedett viselkedés specifikációja.

17. Mik a rendszertervezés alapvető lépései? Ismertesse a rendszertervezés folyamatát!
Követelménydefiníció -> Rendszertervezés -> Alrendszerek tervezése -> Rendszerintegráció -> Üzembe helyezés -> Rendszerevolúció -> Rendszer leépítése.
A rendszertervezés folyamata:
• **A követelmények csoportosítása**: követelményeknek kapcsolódó csoportokra osztása.
• **Alrendszerek meghatározása**: alrendszerek olyan halmazának meghatározása, amelyek együttesen képesek a rendszerkövetelmények teljesítésére.
• **Követelmények hozzárendelése az alrendszerekhez**: nehézségbe ütközik, ha COTS rendszereket integrálunk.
• **Alrendszerek funkcionalitásának specifikálása**.
• **Alrendszerek interfészeinek definiálása**: fontos párhuzamos alrendszer-fejlesztés esetén.

18. Ismertesse a követelmény- és rendszertervezés spirális modelljét!

Probléma definíció -> Követelmény gyűjtés és elemzés -> Architektúra tervezése -> Felülvizsgálat és kiértékelés

19. A rendszermodellezés alapvető eszközei. Blokkdiagram, alrendszerek leírása.
Az architektúra modell a rendszert alkotó alrendszerek absztrakt reprezentációja.
Tartalmazhatja az alrendszerek közötti főbb információfolyamokat is.
Általában **blokk-diagram** formájában használjuk. Pl.: betörésjelző rendszer:

Alrendszerek leírása:

Alrendszer	Leírás
Mozgásérzékelők	A monitorozott helységekben mozgást érzékel
Ajtószenzorok	Érzékeli az épület külső ajtóinak nyílását
Központi egység	A rendszer működését vezérli
Sziréna	Hangjelzést ad behatolás esetén
Hangszintetizátor	Hangüzenetet szintetizál a behatoló feltételezett helyéről
Telefonos hívó	Hívásokat generál pl. a rendőrség, biztonságiak, stb. felé

20. Mi a COTS rendszer?
Már létező vagy készen vásárolható rendszerek.

21. Hogyan történik az alrendszerek fejlesztése?
• Általában párhuzamosan zajlik a hardver, szoftver és a kommunikáció fejlesztése.
• Tartalmazhat COTS (Commercial Off-The-Shelf) rendszerek beszerzését is.
• A fejlesztő csoportok között nincs kommunikáció.
• Amennyiben változtatásra van szükség, a lassú és bürokratikus engedélyeztetési eljárások miatt gyakran határidő módosítás is szükséges.

22. Mi a rendszerintegráció, hogyan történik?
Az a folyamat, amelynek során a hardver, szoftver és személyi állomány együttesen rendszert alkot.
• Célszerű inkrementálisan végezni (egyszerre csak egy alegység integrálása).
• Az alegységek közötti interfész problémák rendszerint ebben a fázisban derülnek ki.
• A rendszerkomponensek koordinálatlan beszállítása gondokat okoz.

23. Ismertesse a telepítés során várható főbb problémákat.
A rendszert elkészülte után a megrendelőnél üzembe kell helyezni.
Problémák:
• A környezettel kapcsolatos feltételezések esetleg tévesek voltak.
• Az új rendszerrel szemben ellenállást tapasztalhatunk a befogadó oldalon.
• A rendszernek egy ideig esetleg együtt kell létezni más rendszerekkel.
• Fizikai problémák is felléphetnek a telepítés során (pl. kábelezési gondok).
• Az operátorok betanításról gondoskodni kell.

24. Mit jelent a rendszerek evolúciója?
Nagy rendszerek hosszú élettartamúak. Lépést kell tartani a változó követelményekkel.
Az evolúció költséges!
• A változásokat technikai és üzleti szempontból is elemezni kell.
• Az alrendszerek egymásra hatása miatt nem várt problémák adódhatnak.
• Ritkán ismertek az eredeti tervezési megfontolások.

• A rendszer struktúrája sérül a folyamatos változtatások során.
Legacy rendszer: az a régi rendszer, amelynek fenntartása elengedhetetlen.

25. Ismertesse a rendszerfejlesztést befolyásoló főbb emberi és szervezeti tényezőket!
Emberi és szervezeti tényezők:
Változások a munkafolyamatban:
• A rendszer bevezetése szükségessé tesz-e változásokat a munkafolyamat során?
Munkahelyek veszélyeztetése:
• Elvesznek-e munkahelyek a rendszer bevezetése miatt, illetve meg kell-e változtatni a jelenlegi munkavégzés módját?
Szervezeti változások:
• Megváltoztatja-e a rendszer a szervezeti egység jelenlegi politikai/hatalmi elrendezését?

26. Ismertesse a beszerzés folyamatát egyedi rendszer és COTS alrendszereket használó rendszer esetén!
Piackutatás: létező szoftverek keresése ->
Ajánlatok (COTS létezik) -> Követelmények adaptálása -> Rendszer kiválasztása -> bekérése -> Szállító kiválasztása
szerződés (Egyedi rendszer) -> Pályázat kiírása -> Tender kiválasztása -> Tárgyalás a feltételeiről -> Fejlesztési szerződés megkötése

27. Ismertesse a Legacy rendszerek legfontosabb tulajdonságait, tipikus előfordulási lehetőségeit.
Legacy rendszerek: olyan ember-gép rendszerek, amelyek régi vagy elavult technológiával lettek kifejlesztve.
Üzleti szempontból kritikus fontosságúak és gyakran túl kockázatos ezek felszámolása vagy cseréje.
Pl.:
• Banki könyvelési rendszerek
• Repülési karbantartó rendszerek
A legacy rendszerek korlátozzák az új üzleti eljárások bevezetését.
A vállalati kiadások jelentős részét ezek emésztik fel.

28. Melyek a szoftvergyártás alapvető tevékenységei?
• **Specifikáció**
• **Tervezés**
• **Ellenőrzés (validáció)**
• **Továbbfejlesztés (evolúció)**

29. Sorolja fel az alapvető szoftvergyártási modelleket!
A vízesés (waterfall) modell
Evolúciós fejlesztési modellek
Komponens alapú fejlesztés
A fenti modellek variációja

30. Ismertesse a vízesés modellt és annak fázisait!

A vízesés modell fázisai:
Követelményanalízis és – definíció
Rendszer- és szoftvertervezés
Implementáció és a részegységek tesztelése
Részegységek integrálása és a rendszer tesztelés
Működtetés és karbantartás
31. Mik a vízesés modell hátrányai? Milyen rendszerek fejlesztése esetén hasznos ez a modell? Milyen rendszerek esetén nem?
A vízesés modell legfőbb hátrányai:
- A gyártás megindulás a után nehéz változásokat beépíteni.
- Egy munkafázisnak be kell fejeződni, mielőtt a következő elkezdődhet.
- Nehéz a változó megrendelői igényekhez igazodni, mert a projekt nehezen változtatható részegységekből áll.

Hasznos, ha a követelmények jól ismertek és csak nagyon kis változások lehetségesek a fejlesztés során.
Sajnos csak kevés üzleti rendszernek vannak stabil követelményei.
Főleg **nagy rendszerek** fejlesztése során használják, ahol a fejlesztés több helyszínen történik.

32. Ismertesse az evolúciós fejlesztés alapelveit és 2 fő formáját!
Kísérletező fejlesztés: cél: a megrendelővel együtt egy kezdeti durva specifikációból a végleges rendszert kialakítani. A biztos követelményekből kiindulva a megrendelő igényei szerint újabb funkciókkal bővíthető a rendszer.
Eldobható prototípus: cél: a homályos követelmények tisztázása. A legkevésbé kiforrott követelményekből indul, hogy tisztázza a valós igényeket.

33. Mik az evolúciós fejlesztés előnyei és hátrányai. Hol alkalmazható jól?
Problémák:
• A fejlesztés nem átlátható
• A rendszerek gyakran rosszul strukturáltak
• Speciális felkészültségre lehet szükség (pl. *rapid prototyping* nyelvek)
Alkalmazhatóság:
• Kis- és középméretű interaktív rendszerek
• Nagy rendszerek részegységei (pl. felhasználói felület)
• Rövid élettartamú rendszerek.

34. Ismertesse a komponens-alapú szoftverfejlesztés alapelveit és menetét!
Szisztematikus újrafelhasználáson alapul.
A rendszereket már létező, vagy készen vásárolható (COTS) rendszerekből integráljuk.
A szoftvergyártás lépései:
Komponens analízis
Követelmények módosítása
Rendszertervezés újrafelhasználással
Fejlesztés és integráció
Egyre szélesebb körben terjed, ahogy a komponens szabványok fejlődnek.

35. Ismertesse az újrafelhasználás-alapú fejlesztés menetét!
Követelmény specifikáció -> Komponens analízis -> Követelmények módosítása -> Rendszertervezés újrafelhasználással -> Fejlesztés és integráció -> A rendszer validációja

36. Miért hasznos megközelítés az iteratív szoftvergyártás? Ismertesse 2 alapvető típusát!
- A rendszerkövetelmények MINDEN projekt során változnak, így az iteratív megközelítés (korábban elvégzett munkafázisok átdolgozása) minden nagyobb rendszer fejlesztésének része.
- Az iteratív megközelítés valamennyi alapvető módszerhez alkalmazható.

2 kapcsolódó megközelítés:
• Inkrementális teljesítés
• Spirális fejlesztés

37. Ismertesse az inkrementális teljesítés alapelveit, menetét, valamint legfőbb előnyeit!
A rendszert nem egy részletben szállítjuk, hanem a fejlesztés és átadás részekre van bontva. Minden újabb átadott részegység a rendszer újabb funkcionalitását valósítja meg.
A felhasználó igényeknek megfelelő prioritási sorrendben szállítunk, a legfontosabb funkciókkal kezdve.
- Amint egy részegység fejlesztése elkezdődött, annak követelményeit „befagyasztjuk". Későbbi részegységek követelményei még változhatnak.

Az inkrementális teljesítés előnyei:
- A rendszer korábban kezdheti meg (rész)működését.
- Korábbi komponensek prototípusként működnek, így a későbbi részegységek követelménytervezésében ezek is segítenek.
- Kisebb a projekt teljes csődjének esélye.

A legfontosabb szolgáltatásokat tesztelik a legtovább.

38. Ismertesse a spirális fejlesztés alapelveit. Mit jelentenek a hurkok és a szektorok?
A gyártási folyamat sokkal inkább egy spirállal jellemezhető, mint tevékenységek (visszalépéses) sorozataként.
A spirál minden hurka a gyártási folyamat egy fázisát jelképezi.
Nincsenek fix hurkok (pl. specifikáció, vagy tervezés). A hurkokat az igényeknek megfelelően alakítjuk ki.
- A kockázatkezelés explicit módon megjelenik a gyártási folyamatban.

A spirális modell szektorai:
- Célkitűzések megállapítása: az adott fázis céljainak megállapítása.
- Kockázatbecslés és –csökkentés: a kockázati tényezők felmérése, valamint a legfőbb kockázati faktorok várható hatásának csökkentése.
- Fejlesztés és validáció: az általános módszerek közül bármely kiválasztása.
- Tervezés: a projekt áttekintése és a spirál következő fázisának megtervezése.

39. Mi a szoftver specifikáció feladata. Ismertesse a követelménytervezés lépéseit és a követelmény-tervezési eljárás menetét!
Választ keresünk a következő kérdésekre: milyen szolgáltatásokat várunk el a rendszertől, milyen kötöttségeket és kényszereket kell figyelembe venni a fejlesztés és üzemeltetés során.
Követelménytervezési lépései:
• **Megvalósíthatósági tanulmány**
• **Követelmények gyűjtése és analízise**
• **Követelmény specifikáció**
• **Követelmény validáció**

40. Ismertesse a szoftvertervezés lépéseit és folyamatát.
A tervezés lépései:
- **Architektúra tervezése**
- **Absztrakt specifikáció**
- **Interfészek tervezése**
- **Komponensek tervezése**
- **Adatstruktúrák tervezése**
- **Algoritmusok tervezés**

A szoftvertervezés folyamata:

41. Ismertesse a hibakeresés folyamatát!

Hiba lokalizálása -> Hibajavítás tervezése -> Hibajavítás -> Program ismételt tesztelése

42. Mi a szoftver validáció célja és elemei?
- A verifikáció és validáció (V & V) célja annak bizonyítása, hogy a rendszer teljesíti a specifikációban foglaltakat és a felhasználó igényeinek megfelelően működik.

Elemei: **ellenőrzés**, **felülvizsgálat** és **rendszertesztelés**.
- Rendszertesztelés: a rendszer futtatása olyan tesztadatokkal, amely a specifikáció szerint a valós működés során előfordulhat.

43. Ismertesse a tesztelési eljárást, illetve a tesztelés egyes lépéseit, ennek illeszkedését a fejlesztési folyamatba.
A tesztelési eljárás:
A tesztelés lépései:
- **Komponens- és részegység-tesztelés**:
  • A különálló komponenseket egymástól függetlenül teszteljük.
  • A komponensek lehetnek: függvények, objektumok, vagy ezek összetartozó csoportjai.
- **Rendszertesztelés**: A rendszer egészének tesztelése. Különösen fontos az eredő tulajdonságok ellenőrzése.
- **Végteszt (átadási teszt)**: A megrendelő által szolgáltatott valós adatokkal annak ellenőrzése, hogy a megrendelő igényeit valóban kielégíti.

44. Mit jelent a szoftver evolúciója? Miért van rá szükség? Mik a legfőbb kiváltó okai?
- Ahogy a változó üzleti-gazdasági körülmények miatt a követelmények változnak, a kiszolgáló szoftvernek is változnia és fejlődnie kell.
- Bár a fejlesztés és karbantartás között régebben éles határvonal húzódott, ez egyre kevésbé releváns, hiszen egyre kevesebb a teljesen új rendszer (evolúció).

45. Ismertesse a Rational Unified Process (RUP) filozófiáját, főbb jellemzőit.
- Korszerű tervezési modell, amely az UML, és a hozzá kapcsolódó eljárásokból jött létre.

Általában 3 nézetet használunk:
• Dinamikus nézet: a fázisokat az idő függvényében mutatja.
• Statikus nézet: a gyártási folyamatokat mutatja.
• Gyakorlati nézet: jól bevált gyakorlati útmutató.

A RUP fázisai:
- Alapozás: a rendszer számára egy üzleti modell megalkotása.
- Kidolgozás: a problématér megértése és a rendszer-architektúra kidolgozása.

Konstrukció: rendszertervezés, programozás és tesztelés.

Átmenet: a rendszer telepítése a működési környezetbe.

46. Mik a számítógéppel segített szoftverfejlesztés által nyújtott legfontosabb szolgáltatások? Soroljon fel tipikus eszközöket!

CASE (Computer-aided software engineering) olyan szoftver, amely a szoftverfejlesztés és evolúció folyamatát segíti.

Tevékenységek automatizálása:

- Grafikus szerkesztők rendszermodellek fejlesztésére.
- Adatkönyvtár tervezési entitások menedzselésére.
- Grafikus felhasználói felületszerkesztő.
- Debuggerek hibakereséshez.
- Automatikus transzlátorok új programverziók generálásához.

47. Mik a CASE eszközök integrációjának alapvető típusai?

Eszköz (tool): elemi műveletek támogatása szolgál.

Munkapad (workbench): egy gyártási fázist támogat. Általában néhány integrált eszközt tartalmaz.

Környezet (environment): az egész szoftvergyártási folyamat minden lényeges elemét tartalmazza. Általában számos integrált munkapadot tartalmaz.

48. Mi a szoftver-projekt menedzsment célja?

Biztosítja, hogy a szoftvert időben és a megadott ütemterv szerint szállítsák, a megrendelő és a fejlesztő szervezetek által állított követelmények betartásával.

49. Miért különleges a szoftvermenedzsment más menedzsment tevékenységekhez képest?

A termék nem materiális.

A termék különlegesen flexibilis.

A szoftvermérnökség nem rendelkezik más mérnöki tudományokhoz hasonló szilárd alapokkal (pl. gépész-, villamosmérnök).

A szoftverfejlesztési eljárás nem szabványosított.

Sok szoftver projekt unikális.

50. Mik a menedzser fő tevékenységei?

Pályázatok írása.

Projekttervezés és ütemezés.

Költségtervezés.

Projekt felügyelet és ellenőrzés.

Személyzet kiválasztása és értékelése.

Beszámolók írása és prezentációja.

51. Mi a projekttervezés?

Valószínűleg a legidőigényesebb projektmenedzsment tevékenység.

Állandó tevékenység a kezdeti koncepciótól a rendszer átadásáig. A terveket állandóan felül kell vizsgálni amint újabb információ áll rendelkezésre.

Több különböző típusú terv készül egy szoftverprojekt során, amelyek az ütemezéssel és a pénzügyekkel foglalkoznak.

52. Sorolja fel a projekttervek főbb típusait és röviden ismertesse ezeket!

Minőségbiztosítási terv: a projekt során használatos minőségbiztosítási eljárások és szabványok leírása.

Validációs terv: a rendszervalidáció során használt megközelítés, a felhasznált erőforrások és ütemterv leírása.

Konfigurációs és menedzsment terv: konfiguráció menedzsment eljárások és az alkalmazott struktúrák leírása.

Karbantartási terv: a rendszer karbantartási követelményeit becsli (karbantartási költség és egyéb ráfordítás).

Továbbképzési terv: A projekten dolgozók szakmai felkészültségének és tapasztalatának fejlesztési terve.

53. Ismertesse a projekttervezés folyamatát (pl. pszeudo-kóddal)!

A projektet érintő kényszerek és megszorítások definiálása.

A projekt paraméterek kezdeti becslése.

A projekt határidők és teljesítések definiálása

while a projekt nem kész és nem ment csődbe **loop**
  Projekt ütemterv felvázolása
  Az ütemterv szerinti tevékenységek indítása
  Wait (egy ideig)
  A projekt előmenetelének felülvizsgálata
  A projekt becsül paramétereinek felülvizsgálata
  Projekt ütemterv frissítése
  A kényszerek és teljesítések újratárgyalása
  if (probléma adódik) **then**
    Technikai felülvizsgálat, esetleg revízió kezdeményezése
  end if
end loop

54. Mi a projektterv feladata, milyen főbb információkat tartalmaz? Ismertesse a projekttery tipikus felépítését.

A projektterv tartalmazza:

- A projekt számára elérhető erőforrásokat.
- A munka felosztását.
- A munka ütemtervét.

A projektterv felépítése:

- Bevezetés.
- A projekt felépítése.
- Kockázatelemzés.
- Hardver és szoftver erőforrás igények.
- A munka felosztása.
- A projekt ütemterve.
- Felügyeleti és beszámolási mechanizmusok.

55. Ismertesse a határidők (mérföldkövek) és teljesítések definícióját!

Határidő (mérföldkő): egy tevékenységi szakasz lezáró pontja.

Teljesítés: a megrendelőnek leszállított és átadott eredmény.

56. Ismertesse a projekt ütemezés feladatát és menetét! Ismertesse a grafikus reprezentációk fajtáit (aktivás hálózat, aktivás időtdiagram (oszlopdiagram), munkaerő hozzárendelés) és felhasználási módjukat!

Feladata:

Projekt szakaszokra osztása.

Minden szakaszra az időigény és a szükséges erőforrások becslése.

Szakaszok párhuzamos ütemezése a munkaerő optimális kihasználásával.

A szakaszok közötti függések minimalizálása, az egymásra váró szakaszok miatti késések elkerülése érdekében.

A projektmenedzser intuíciójától és tapasztalatától függ.

A projektütemezés menete:

Az oszlopdiagram és az aktivás hálózatok:

A projekt ütemezés grafikus reprezentációi.

Mutatja a szakaszokra bontást. a szakaszok ne legyenek túl rövidek (legalább 1-2 hét).

A tevékenységdiagram a szakaszok függőségeit és a kritikus útvonalat is mutatja.

Az oszlopdiagram (bar chart) az ütemezést naptárszerűen mutatja.

57. Mi a kockázatkezelés feladata? Mi a kockázat? Mi a szoftver-kockázatok három alapvető fajtája? Soroljon fel példákat az egyes kockázatokra!

Kockázatkezelés: a kockázatok felmérésével és azok projektre gyakorolt hatásának minimalizálásának tervezésével foglalkozik.

Kockázat: annak a valószínűsége, hogy egy kellemetlen körülmény bekövetkezik.

- **Projektkockázatok**: a határidőket és az erőforrásokat befolyásolják.
- **Termékkockázatok**: a fejlesztett szoftver minőségét és teljesítményét befolyásolják.
- **Üzleti kockázatok**: a beszerző, illetve fejlesztő céget befolyásolják.

58. Ismertesse a kockázatkezelés menetét, annak 4 fő lépését!

Kockázatok azonosítása: a projekt-, termék-, és üzleti kockázatok azonosítása.

Kockázat analízis: a fenti kockázati tényezők valószínűségének becslése.

Kockázattervezés: a kockázatok hatásának minimalizálása érdekében tervek felvázolása.

Kockázatfigyelés: a kockázatok figyelése a projekt teljes időtartama alatt.

59. Mi a kockázat-azonosítás során azonosított fő (6) kockázat-típus?

Technológiai kockázatok.

Személyi kockázatok.

Szervezeti kockázatok.

Eszköz kockázatok.

Követelmény kockázatok.

Becslési kockázatok.

60. Mi a kockázatanalízis feladata és menete?

Minden kockázati tényező valószínűségének és komolyságának felmérése.

Valószínűség lehet: nagyon alacsony, alacsony, közepes, magas, nagyon magas.

Hatás lehet katasztrofális, komoly, elviselhető, jelentéktelen.

61. Mi a kockázattervezés feladata és főbb statégiái?

Minden kockázati tényező kezelésére stratégia kidolgozása.

Elkerülési stratégiák: a kockázati esemény bekövetkezési valószínűségét csökkentjük.

Minimalizáló stratégiák: a kockázati tényező hatását a projektre vagy a termékre csökkentjük.

Vészterv: ha a kockázati esemény bekövetkezik, a vészterv kezeli.

62. Mi a kockázatfigyelés feladata és menete?

Időközönként minden kockázati tényező vizsgálata:

- Valószínűsége nő vagy csökken?
- A kockázati tényező hatása változott?

A menedzsment projektmegbeszélésein az összes kockázati tényező megvitatása.

63. Mi a követelménytervezés, mik a követelmények?

Követelménytervezés: A felhasználói igények és a rendszer működési feltételek feltárásának folyamata.

Követelmények: a rendszerről a követelménytervezés folyamata során leírt szolgáltatások és feltételek/kényszerek halmaza.

64. Mi a követelmény? Két fő típusa.

Követelmények: a rendszerről a követelménytervezés folyamata során leírt szolgáltatások és feltételek/kényszerek halmaza.

Típusai:

- **Felhasználói követelmények**
- **Rendszerkövetelmények**

65. Mik a felhasználói követelmények és a rendszer követelmények?

Felhasználói követelmények: a rendszer szolgáltatásairól és a működési feltételekről szóló természetes nyelven írt állítások és diagramok. Az ügyfél számára készül.

Rendszerkövetelmények: strukturált dokumentum, amely tartalmazza a rendszer funkcióinak, szolgáltatásainak és működési feltételeinek részletes leírását. Definiálja az implementálandó feladatokat, így a megrendelő és a szállító közti szerződés része lehet.

66. Mik a funkcionális, nem funkcionális és környezeti (domain) követelmények?

Funkcionális követelmények: milyen szolgáltatásokat kell a rendszernek nyújtania, hogyan kell bizonyos bemeneti adatokra reagálnia és hogyan kell viselkednie egyes helyzetekben.

Nem funkcionális követelmények: a rendszer szolgáltatásaira és funkcióira vonatkozó feltételek és kényszerek.

Környezeti (domain) követelmények: olyan követelmények, amelyek a felhasználói környezetből erednek és ennek a környezetnek a sajátságait tükrözik.

67. Mik a funkcionális követelmények fő jellemzői és fajtái? Milyen elvárásaink vannak a funkcionális követelményekkel szemben?

Leírja a rendszer szolgáltatásait.

Függ a szoftver típusától, a várható felhasználói körtől és attól, hogy hol fogják a szoftvert használni.

A funkcionális felhasználói követelmények magas szintű állítások is lehetnek a rendszer elvárt viselkedéséről, de a **funkcionális rendszerkövetelmények**nek a szolgáltatások részletes leírását kell tartalmaznia.

68. Mik a nem funkcionális követelmények fő jellemzői és típusai?

A rendszer-követelményeket és a működési feltételeket, kényszereket definiálják.
Pl.: megbízhatóság, válaszidő, tárolásra vonatkozó követelmények. Kényszerek lehetnek pl. az I/O eszközök adottságai, adatformátumok, stb.

Fejlesztésre vonatkozó követelményeket is meg lehet fogalmazni: Pl. egy adott CASE eszköz, programozási nyelv, vagy fejlesztési módszer használata.

A nem funkcionális követelmények sokszor kritikusabbak, mint a funkcionális követelmények. Ha ezek nem teljesülnek, a rendszer használhatatlan.

A nem funkcionális követelmények típusai:

Termék követelmények: olyan követelmények, amelyek meghatározzák a termék viselkedési módját.
Szervezeti követelmények: a szervezet stratégiájából és működési módjából következő követelmények.
Külső követelmények: a rendszer és a fejlesztési eljárás szempontjából külső tényezők hatására fellépő követelmények.

69. Mit jelent a nem funkcionális követelmények ellenőrizhetősége? Miért nehéz probléma ez? Adjon példát lehetséges mértékekre!
A nem funkcionális követelményeket nehéz precízen megfogalmazni, a nem precíz követelményeket pedig nehéz ellenőrizni.
Cél: a felhasználó általános és átfogó szándéka. Pl. könnyű használhatóság.
Ellenőrizhető nem funkcionális követelmény: olyan követelmény, amely objektíven mérhető mértéket tartalmaz.

Követelmények mértékei lehetnek:
Sebesség: másodpercenkénti tranzakciók száma, válaszidő, képernyő-frissítési idő.
Méret: megabájt, ROM lapkák száma.
Könnyű felhasználhatóság: betanítás idő, Súgó lapok száma.
Megbízhatóság: Meghibásodások közötti átlagos idő (*mean time to failure*, MTF), leállás valószínűsége, hibagyakoriság, rendelkezésre állás.
Robusztusság: újraindítási idő hiba után, események hány százaléka okoz hibát, adatvesztés valószínűsége hiba esetén.
Hordozhatóság: a platformfüggő utasítások aránya, támogatott platformok száma.

70. Mit jelent a követelmények egymásra hatása? Adjon példát nem funkcionális követelmények közötti konfliktusokra!
Komplex rendszerekben gyakori a különböző nem funkcionális követelmények közötti konfliktus.
Példa: repülőgép irányítási rendszere:
- A minél kisebb súly elérése érdekében a rendszerben használt lapkák számát minimalizálni kell.
- A teljesítmény-felvétel csökkentése érdekében alacsony fogyasztású lapkákat kell használni.
- Az alacsony fogyasztású lapkákból több (darab) kell. Melyik a fontosabb követelmény?

71. Mik a környezeti (domain) követelmények, főbb típusai, azonosításuk problémái?
Az alkalmazás környezetéből származtatható. A környezetből adódó rendszertulajdonságokat írja le.
Lehetnek új funkcionális követelmények, létező követelményekhez újabb kényszerek, vagy specifikus számításokat definiálhatnak.
Ha a környezeti követelményeket nem elégíti ki a rendszer, akkor teljesen használhatatlan lehet.

72. Hogyan történik a felhasználói követelmények megfogalmazása? Milyen problémákat vet fel a természetes nyelvek használata?
Funkcionális és nem funkcionális követelményeket fogalmaz meg oly módon, hogy az a rendszer (mélyebb technikai ismeretekkel nem rendelkező) felhasználói is megértsék.
A felhasználói követelményeket természetes nyelven, valamint táblázatok és ábrák segítségével, a felhasználók számára érthető módon kell megfogalmazni.
A természetes nyelvek problémái:
Nem világos: nehéz precíznek lenni úgy, hogy a dokumentum ne váljon nehezen olvashatóvá.
Követelmények keveredése: funkcionális és nem funkcionális követelmények könnyen keveredhetnek.
Követelmények összeolvadása: különböző követelmények együtt fogalmazódnak meg.

73. Hogyan írjuk le a felhasználói követelményeket?
Legyen egy állandó formátumunk a követelmények számára.
Használjuk a kifejezéseket következetesen.
Pl. „kell" a szükséges, a „javallott" a kívánatos követelmények jelzésére.
Használjunk szövegkiemeléseket a követelmény fontos részeinek jelzésére.
Ne használjunk számítógépes zsargont.

74. Hogyan történik a rendszerkövetelmények leírása? A természetes nyelvi specifikáció problémái, ennek alternatívái.
A rendszer funkcióinak, szolgáltatásainak és működési feltételeinek a felhasználói követelményeknél részletesebb leírása.
Ez lesz a rendszertervezés alapja.
A szerződés része lehet.
A rendszerkövetelményeket definiálhatjuk vagy illusztrálhatjuk különféle rendszermodellekkel.
A természetes nyelvi specifikáció problémái:
Többértelműség: a követelmények írói és olvasói ugyanazt kell értsék a szavakon.
Túlságos flexibilitás: ugyanazt a dolgot sokféleképpen lehet elmondani.
A modularizálhatóság hiánya: a természetes nyelvi struktúrák nem alkalmasak a rendszerkövetelmények strukturálására.
A természetes nyelvi specifikáció alternatívái:
Strukturált természetes nyelv: a specifikáció leírására standard formátumok és sablonok használata.
Terv-leíró nyelvek: a specifikációt a rendszer egy működési modelljének segítségével adja meg, egy programnyelv szerű, de absztraktabb nyelv segítségével.
Grafikus jelölések: a rendszer funkcionális követelményeit egy szöveges megjegyzésekkel bővített grafikus nyelv segítségével írja el. Korai példa: SADT. Manapság a use-case leírások és a sorrend (sequence) diagramok használatosak.
Matematikai specifikáció: matematikai elveken (pl. véges állapotú automaták, halmazok) alapuló leírási módok.

75. Mik a strukturált nyelvi specifikációk? Form-alapú és táblázatos módszerek.
- A követelmény-író szabadságát előre definiált követelmény-sablonok korlátozzák.
- Minden követelményt egy standard módon írunk meg.
- A leírásban használható terminológia korlátozva lehet.
- Előny: a természetes nyelv kifejező ereje megmarad, de mégis egy egységes forma alakítható ki.

Űrlap (form)-alapú módszer:
- A funkció vagy entitás definíciója.
- A bementek leírása + honnak erednek.
- A kimenetek leírása + hová mennek.
- Más felhasznált entitások felsorolása.
- Elő- és utófeltételek (*pre-, post-condition*).
- Mellékhatások leírása.

Táblázatos módszer:
- Természetes nyelvek kiegészítésére.
- Különösen hasznos, amikor alternatív végrehajtási módokat definiálunk.

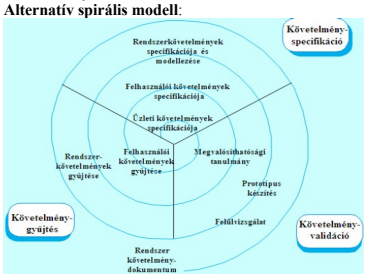
76. Milyen a grafikus modellek alkalmazhatók funkcionális követelmények leírására?
A grafikus modellek akkor hasznosak, amikor állapotok változását, vagy tevékenységek sorozatát kell leírni.
Szekvencia diagramok:
- Események sorozatát mutatja a rendszerben valamely felhasználói interakció során.
- Felülről lefelé olvasandó.

77. Mi az interfész specifikáció feladata, milyen interfész-típusokat használunk?
A legtöbb rendszernek más rendszerekkel együtt kell működnie és az interfészeket a követelmények részeként specifikálni kell.
Interfész típusok:
- **Procedurális interfészek**
- **Adatstruktúrák**
- **Adatreprezentációk**

78. Mi a követelmény-dokumentum, mit tartalmaz? Kik számára készül? Milyen struktúrát ajánl a dokumentum számára az IEEE szabvány?
Követelmény-dokumentum: egy hivatalos dokumentum, amely tartalmazza, hogy mit várunk a rendszer fejlesztőitől.
Tartalmazza a felhasználó követelményeket és a rendszerkövetelmények specifikációját.
Ez nem terv. Amennyire lehetséges, azt tartalmazza, hogy a rendszernek MIT kell csinálni, nem pedig azt, hogy HOGYAN.
Felhasználói:
- Megrendelő, Menedzserek, Rendszertervezők, Tesztmérnökök, Üzemeltetők és karbantartók

IEEE szabvány struktúrajavaslata:
- Bevezetés.
- Általános leírás.
- Az egyes követelmények leírása.
- Függelékek.
- Index.

79. Ismertesse a követelménytervezési eljárás folyamatát és alternatív spirális modelljét!
Követelménytervezési eljárás folyamata:
Követelmények gyűjtése, Követelmények analízise, Követelmények validálása, Követelmény-menedzsment.
Alternatív spirális modell:



80. Mi a megvalósíthatósági tanulmány, mi a célja, főbb tulajdonságai?
A megvalósíthatósági tanulmány dönti el, hogy érdemes-e a rendszert fejleszteni.
Rövid, célirányos tanulmány arról, hogy:
- hozzájárul-e a rendszer a szervezet célkitűzéseinek eléréséhez.
- a rendszer a jelenlegi technológiával és pénzügyi kerettel megvalósítható-e.
- a rendszer integrálható-e a jelenleg használatos többi rendszerrel.

Információ gyűjtés, értékelés, jelentés írása.

81. Hogyan történik a követelménytervezés során az információgyűjtés és –analízis? Mik a fő nehézségek?
Követelmény-becslésnek vagy -feltárásnak is hívjuk.
A műszaki szakemberek a megrendelővel az alkalmazási környezet, a kívánt rendszerszolgáltatások és a működési feltételek feltárásán dolgoznak.
A követelményanalízis problémái:
- A részvényesek nem tudják, valójában mit szeretnének.
- A részvényesek követelményeiket saját nyelvezetükön fogalmazzák meg.
- Különböző részvényeseknek ellentmondó követelményei lehetnek.
- A rendszerkövetelményeket szervezeti és politikai tényezők is befolyásolhatják.
- A követelmények változnak az analízis során: új részvényesek bukkanhatnak fel, illetve az üzleti környezet is változhat.

82. Ismertesse a követelmény-spirál felépítését és a 4 fő tevékenységet!
Követelmények feltárása -> Követelmények osztályozása és szervezése -> Követelmény-prioritások felállítása. Tárgyalások -> Követelmények dokumentálása
- **Követelmények feltárása**: a részvényesekkel való interakció során fel kell fedni igényeiket. Az alkalmazási környezet követelményeit is ebben a fázisban kell feltárni.
- **Követelmények osztályozása és szervezése**: a kapcsolódó követelmények csoportosítása és koherens rendszerbe szervezése.
- **Prioritások, tárgyalások**: a követelmények fontossági sorrendbe állítása. A konfliktusok feloldása.
- **Követelmények dokumentálása**: a követelmények dokumentálása. Ez lesz a spirál következő körének bemenete.

83. Ismertesse a követelmények feltárásának célját és módszereit! Kiket nevezünk részvényeseknek?
Cél: információgyűjtés a javasolt és a jelenlegi rendszerről, majd ebből a felhasználói- és rendszerkövetelmények leszűrése.
Információforrások lehetnek:
- dokumentáció
- részvényesek
- hasonló rendszerek specifikációi.

Részvényesek: az információgyűjtés és –analízisben a végfelhasználók, menedzserek, stb.

84. Mik a nézőpontok, típusok és szerepük a követelménytervezésben?
Nézőpontok: lehetőséget adnak a a követelmények strukturálására, a különböző részvényesek szempontjainak reprezentálására.
A részvényesek különböző nézőpontokba sorolhatók.
Fontos a több szempontból történő elemzés.
Nincs egyetlen helyes módja a rendszerkövetelmények analízisének.
A nézőpontok típusai:
- **Interaktív nézőpontok**: emberek, vagy más rendszerek, amelyek kölcsönhatásban vannak a rendszerrel.
- **Indirekt nézőpontok**: olyan részvényesek, akik nem használják a rendszert, de a követelményeket befolyásolják.
- **Alkalmazási környezet (domain) nézőpontok**: alkalmazási környezet jellemzői és kényszerei, amelyek befolyásolják a követelményeket.

85. Mik az interjúk, ezek típusai és a hatékony interjúkészítés feltételei?
Formális vagy informális interjúk keretében a részvényeseknek kérdéseket teszünk fel a rendszerről, amit használnak, és a rendszerről, amit fejlesztünk.
Az interjúk típusai:
- **Zárt**: egy előre meghatározott kérdés-csoportra kell válaszolni.
- **Nyílt**: nincs előre meghatározott menetrend, a megválaszolandó problémákat a részvényesekkel együtt tárjuk fel.

A hatékony interjú feltételei:
- Az interjú készítője legyen elfogulatlan, figyeljen a részvényesekre és ne legyenek prekoncepciói a követelményekről.
- Legyenek felteendő kérdések vagy javaslatok a meginterjúvoltak számára, ne várjuk, hogy hasznos információt adnak a „mit szeretne" kérdésre.

86. Mik a szcenáriók (forgatókönyvek), mit tartalmaznak?
Szcenáriók (forgatókönyvek): valós életből vett példák arról, hogy hogyan kell a rendszert használni.
Tartalmazniuk kell:
- A kiinduló szituáció leírását.
- Az események normál menetének leírását.
- Annak leírását, hogy mi sikerülhet rosszul, kivéte.
- Információt már párhuzamos tevékenységekről.
- A szcenárió befejezése utáni állapot leírását.

87. Mik az esettanulmányok (use cases), mit tartalmaznak? A részletesebb kiegészítő információk közlésének lehetséges módszere: szekvencia-diagram.
Szcenárió-alapú technika, az UML része.
Azonosítja az interakcióban részt vevő aktorokat és leírja magát az interakciót is.
Esettanulmányokkal valamennyi lehetséges, a rendszerrel kapcsolatos interakciót le kell írni.
Szekvencia-diagramok: részletes információkat csatolhatnak az esettanulmányhoz.
Bemutatják az események kezelésének sorrendjét a rendszerben.

88. Mi az etnográfia célja, szerepe? A célzott etnográfia.
Etnográfia: társadalomkutatók foglalkoznak emberek munka közbeni megfigyelésével és ennek analízisével.
Célja:
- A dolgozóknak így nem kell szóban elmagyarázni a munkájukat.
- Fontos társadalmi és szervezeti tényezőkre derülhet így fény.
- Etnográfiai kutatások szerint a munkafolyamatok általában sokkal gazdagabbak és bonyolultabbak annál, mint azt az egyszerű rendszermodellek mutatják.

Célzott etnográfia:
- Légiirányítók munkáját tanulmányozó projektből ered.
- Kombinálja az etnográfiát a prototípus-készítéssel.
- A prototípus-készítés rávilágít a megválaszolatlan kérdésekre és fókuszálja az etnográfiai kutatást.
- Gond az etnográfiával: a jelen gyakorlatot vizsgálja, ami valamilyen, esetleg már nem is releváns történelmi alapokon nyugszik.

89. Mi a követelmény-validáció célja?
Célja: annak igazolása, hogy a követelmények azt tartalmazzák, amit a megrendelő valóban akar.

90. Milyen szempontok szerint kell a követelményeket ellenőrizni?
- **Érvényesség**: a rendszer a megrendelő igényeit legjobban kielégítő szolgáltatásokat nyújtja?
- **Konzisztencia**: vannak a követelmények között ellentmondások, konfliktusok?
- **Teljesség**: a megrendelő számára minden szükséges funkció rendelkezésre áll?
- **Realitás**: a jelenlegi technológiával és költségvetéssel implementálható a rendszer?
- **Verifikálhatóság**: ellenőrizhetők a követelmények?

91. Milyen technikák alkalmazhatók a követelmények ellenőrzésére?
- **Követelményszemle**: a követelmények szisztematikus kézi ellenőrzése.
- **Prototípus készítése**: a rendszer végrehajtható modelljének segítségével ellenőrizzük a követelményeket.
- **Tesztek készítése**: tesztelhetőség ellenőrzése követelmény-tesztek kidolgozásával.

92. Ismertese a követelményszemlék menetét és a fő ellenőrző pontokat!
- **Verifikálhatóság**: a követelmény reálisan tesztelhető?
- **Érthetőség**: mindenki helyesen érti a követelményeket?
- **Követhetőség**: a követelmény eredete világosan meg van fogalmazva?
- **Változtathatóság**: a követelmény megváltoztatható-e más követelményekre gyakorolt nagyobb hatás nélkül?

93. Mi a követelmény menedzsment szerepe? Miért van rá szükség?
A változó követelmények kezelésének folyamata a követelménytervezés és a rendszer fejlesztése során.
A követelmények nem teljesek és nem konzisztensek:
- Új követelmények bukkannak elő, ahogy az üzleti érdekek változnak, vagy a rendszerről egyre teljesebb tudásanyag áll elő.
- Különféle nézőpontoknak más és más követelményei vannak, ezek gyakran egymásnak ellentmondanak.

94. Milyen terveket kell készíteni a követelmények menedzsmentje során?
- **Követelmények azonosítása**: hogyan lesznek az egyes követelmények azonosítva.
- **Változáskövetési eljárás**: ezt az eljárást kell követni követelményváltozás elemzése során.
- **Követési stratégiák**: milyen és mennyi információt kell tárolni a követelmények közötti összefüggésekről.
- **CASE eszköz**: milyen CASE segítség kell a követelményváltozások menedzseléséhez.

95. Milyen információkat kell tárolni a változások követhetősége céljából?
Forrás követés: a követelményeket azokhoz a részvényesekhez köti, akiktől a javaslat származik.
Követelmény követés: egymástól függő követelmények közötti kapcsolatot kezeli.
Tervezés követés: kapcsolatok a követelmények és a terv elemei között.

96. Mik a követelmény-változás menedzsment fő lépései?
- **Probléma-analízis**: a követelményekkel kapcsolatos problémák megvitatása és javaslat a változtatásra.
- **Változás-analízis és költségbecslés**: a változtatás hatásának becslése más követelményekre.
- **Változtatás végrehajtása**: a követelmény-dokumentum és más kapcsolódó dokumentumok megváltoztatása.

97. Mi a rendszermodellezés célja? Főbb modell-típusok.
Célja: a rendszer modellezése egyrészt segíti a rendszerelemzőt a rendszer funkcionalitásának megértésében, másrészt modellek segítségével a megrendelővel is lehet kommunikálni.
Modell-típusok:
- **Adatfeldolgozás modell**: az adat feldolgozásának lépései különböző szinteken.
- **Kompozíció (aggregáció) modell**: egyes entitások hogyan épülnek fel más entitásokból.
- **Architektúrális modell**: a legfontosabb alrendszereket mutatja be.
- **Osztály-modell**: Az entitások közös jellemzőit mutatja be.
- **Gerjesztés/válasz modell**: a rendszernek különféle eseményekre adott reakcióit mutatja.

98. Mi a kontextus modell, mi a szerepe?
Milyen modell-típusokat használhatunk erre a célra?
A **kontextus modellek** a rendszer működési környezetét mutatják be.
Szerepe: architektúrális modellekkel a rendszert és annak más rendszerekkel való kapcsolatát mutatjuk be.
Modell-típusok:
- **Folyamat modellek**
- **Viselkedési modellek**

99. Mit tartalmaznak a folyamat modellek?
Folyamat modellek: a teljes folyamatot, és ezen belül a rendszer által támogatott folyamatokat írják le.
Az adatfolyam modellek a folyamatokat és a folyamatok közötti információ-áramlást mutatják.

100. Mire szolgálnak a viselkedési modellek, milyen típusai vannak?
A viselkedései modellek a rendszer viselkedését írják le.
Típusai:
- **Adatfeldolgozó modellek**: az adatok feldolgozásának leírása a rendszeren való áthaladásuk során.
- **Állapotgép modellek**: a rendszer eseményekre való válaszát írják le.

101. Milyen az adatfeldolgozó modellek felépítés, felhasználási területei?
Adatfolyam-diagramok jól használhatók a rendszer adatfeldolgozó funkcióinak leírásához.
A feldolgozás lépéseit mutatják, ahogy az adat áthalad a rendszeren.
Az adatfolyam-diagramok számos analízis módszer lényeges részét alkotják.
Egyszerű és intuitív jelölés, a megrendelő is megérti.
Az adat feldolgozását mutatja a bementettől a kimenetig.
Használhatók még a rendszer és annak környezetében lévő más rendszerek közötti adatcsere bemutatására.

102. Mire szolgálnak az állapotgép modellek?
A rendszer viselkedését modellezik annak különböző külső és belső eseményekre adott válaszain keresztül.
Állapotgépek: a rendszer állapotai a csomópontok, köztük futó irányított élek pedig az események. Egy esemény bekövetkezésekor a rendszer egyik állapotból a másikba megy át.
Az állapot-diagramok (statechart) az UML fontos részét alkotják.
Ezzel állapotgépeket írhatunk le.

103. Ismertesse az állapot-diagramok felépítését, elemeit és szabályait!
A modellnek kisebb részekre, al-modellekre bontását teszi lehetővé (dekompozíció).
Az akciók rövid leírása az egyes állapotokban a 'do' utasítás után található.
Kiegészíthető az állapotok és a gerjesztések leírását tartalmazó táblázatokkal.

104. Mire szolgálnak a szemantikus adatmodellek?
Mik az entitás-reláció-attribútum diagramok?
A rendszer által feldolgozott adatok logikai struktúráját írja le.
Entitás–Reláció–Attribútum diagram: a rendszerben használt entitásokat, az ezek közötti relációkat és az entitások attribútumait mutatja be.
Adatbázisok modellezésénél széles körben használt.
Az UML nem nyújt specifikus jelölésrendszert, de a az objektumok és az asszociációk használhatók e célra.

105. Mik az adat-szótárak? Előnyei?
Adat-szótár: a rendszer-modellekben használt valamennyi név listája.
Az entitások, kapcsolatok és attribútumok leírását is tartalmazza.
Előnyei:
- Támogatja a név-menedzsmentet és segít az ütközések elkerülésében.
- Fontos szervezési tudástár: összeköti az elemzés, tervezés és implementáció során összegyűlt információkat.

Sok CASE munkapad támogatja az adat-szótárak kezelését.

106. Mik az objektum modellek? Lehetséges fajtái?
Objektum modellek: a rendszert az objektum-osztályok és ezek kapcsolatán keresztül mutatják be.
Egy objektumosztály egy olyan objektumhalmaz absztrakciója, amelynek elemeinek közös attribútumai és szolgáltatásai (operációi) vannak.
Lehetséges fajtái:
- **Öröklési modellek**
- **Aggregáció modellek**
- **Interakció modellek**

107. Ismertesse az öröklési modellek célját, felépítését, UML-beli reprezentációját!
Egyszeres és többszörös öröklés.
Célja: az objektum-osztályok hierarchiába szervezésére.

Felépítése: a hierarchia csúcsán lévő osztályok az összes osztály közös tulajdonságait reprezentálják. Az objektum-osztályok az attribútumaikat és szolgáltatásaikat egy vagy több szuper-osztálytól öröklik.

UML-jelölések:
• Az objektum-osztályokat téglalapok jelképezik, amelyben a név felül, a tulajdonságok középen, az operációk pedig az alsó részben helyezkednek el.
• Az objektumok közti relációkat (asszociációkat) az objektumokat összekötő vonalak jelképezik.
• Az öröklést itt általánosításnak nevezik és a hierarchiában nem „lefelé", hanem „felfelé" olvasandó.
Többszörös öröklés:
A többszörös öröklés lehetővé teszi objektum-osztályoknak attribútumok és szolgáltatások több (és nem csak egy) szuper-osztálytól való öröklését.

108. Ismertesse az objektum aggregációs modell feladatát, felépítését, jelölését az UML-ben!
Aggregációs modell: azt mutatja, hogy összetett osztályok hogyan állnak össze más osztályokból.
Az aggregációs modellek hasonlóak a szemantikus adatmodellek „tartalmaz" relációjához.
Felépítés és UML-jelölés:
• Az objektum-osztályokat téglalapok jelképezik, amelyben a név felül, a tulajdonságok középen, az operációk pedig az alsó részben helyezkednek el.
• Az objektumok közti relációkat (asszociációkat) az objektumokat összekötő vonalak jelképezik.

109. Milyen modelleket használunk a viselkedés modellezésére?
Viselkedési modellek: az objektumok közötti interakciókat mutatják be a rendszer valamely funkciója során, amelyet esettanulmány (use case) specifikál.
Modell:
Szekvencia-diagramok: az UML-ben az objektumok közötti interakciók modellezésére használjuk.

110. Mik a strukturált módszerek ismérvei, előnyei, hátrányai? Milyen CASE eszközök állnak a rendelkezésre?
A strukturált módszerek fontos eleme a rendszermodellezés.
Ezek a módszerek definiálnak egy modellhalmazt, egy eljárást ezen modellek meghatározására, valamint a modellekre vonatkozó szabályokat és ajánlásokat.
Hátrányok:
• Nem modellezik a nem funkcionális rendszerkövetelményeket.
• Általában nem tartalmaznak arról információt, hogy a módszer alkalmazható-e az adott problémára.
• Túl sok dokumentációt eredményeznek.
• A rendszermodellek néha túl részletesek és a felhasználók számára nehezen érthetőek.
CASE eszközök:
• Diagram szerkesztők
• Modell analízis és ellenőrző eszközök
• Adattár és kapcsolódó kereső nyelv
• Adat-szótár
• Riport definíciós és generátor eszközök
• Sablon definíciós eszközök
• Import/export transzlátorok
• Kódgenerátor eszközök

111. Mik az objektum-orientált tervezés elemei? Mivel foglalkozik az OOA, OOT és az OOP?
Objektum-orientált Analízis (OOA), **Tervezés** (OOT) és **Programozás** (OOP).
OOA: a felhasználói környezet modelljének kidolgozásával foglalkozik.
OOT: a követelményeket kielégítő rendszer modelljének kidolgozásával foglalkozik.
OOP: az OOT realizálásával foglalkozik egy OO nyelv (pl. Java, C++) segítségével.

112. Mik az OOT jellemzői, előnyei?
Az OOT jellemzői:
• Az objektumok a való világ entitásainak reprezentációi, amelyek önmagukat menedzselik.
• Az objektumok önállóak és saját, a külvilág számára közvetlenül nem látható állapottal rendelkeznek.
• A rendszer funkcionalitását objektumok szolgáltatásaiként reprezentáljuk.
• Közös adatterületek nem léteznek. Az objektumok üzenetekkel kommunikálnak.
• Az objektumok lehetnek elosztottak, végrehajtásuk lehet szekvenciális vagy párhuzamos.
Az OOT előnyei:
• Könnyű kezelhetőség. Az objektumok önálló entitásokként foghatók fel.
• Az objektumok potenciálisan újrafelhasználható komponensek.
• Sok rendszerben a való világ entitásai könnyen és értelemszerűen képezhetők le a rendszer objektumaira.

113. Mik az objektumok és az objektum-osztályok? UML jelölések.
Objektumok: a szoftver rendszer entitásai, amelyek a való világ és a rendszer entitásait reprezentálják.
Objektum-osztályok: objektumok sablonjai. Belőlük objektumok hozhatók létre.
Az objektum-osztályok más objektum-osztályoktól attribútumokat és szolgáltatásokat örökölhetnek.
UML-jelölés:
• Az objektum-osztályokat téglalapok jelképezik, amelyben a név felül, a tulajdonságok középen, az operációk pedig az alsó részben helyezkednek el.

114. Ismertesse az objektumok kommunikációjának elvi és gyakorlati lehetőségeit. Az üzenetek implementálásának lehetőségei.
Elvileg: az objektumok üzeneteken keresztül kommunikálnak.
Üzenetek:
• A hívó objektum által kért szolgáltatás neve.
• A szolgáltatás végrehajtásához szükséges információ másolata, valamint az eredmény tárolójának neve.
Gyakorlatban: az üzenteket gyakran eljárás-hívással implementáljuk:
• Név = eljárás neve.
• Információ = paraméter-lista.

115. Mi az általánosítás és az öröklés kapcsolata? UML jelölés. Az öröklés szabályai, előnyei és problémái.
Az objektumok azon osztályok tagjai, amelyek az attribútumait és az operációt definiálják.
Az osztályok egy osztály-hierarchiába szervezhetők, ahol egy osztály (szuper-osztály) egy vagy több osztály (al-osztály) általánosítása lehet.
Az al-osztály örökli a szuper-osztály attribútumait és operációit, valamint saját metódusokat és attribútumokat adhat ezekhez.
Az UML-beli általánosítást az OO nyelvek öröklésként implementálják.
UML-jelölés:
• Az objektum-osztályokat téglalapok jelképezik, amelyben a név felül, a tulajdonságok középen, az operációk pedig az alsó részben helyezkednek el.
• Az objektum-osztályok közt alulról (alosztály) felfelé (szuperosztály vagy alosztály) nyilak jelölik az általánosítást.
Az öröklés előnyei:
• Egy absztrakciós mechanizmus, amely entitások osztályozására használható.
• Egy újrafelhasználási mechanizmus, amely a tervezés és programozás szintjén is használható.
• Az öröklési gráf alkalmazási környezetek és rendszerek szerveződéséről szolgáltat információt.
Az öröklés problémái:
• Az objektum-osztályok nem „önjáróak". A megértésükhöz szükséges a szuper-osztályuk ismerete is.
• A tervezők gyakran újrahasznosítják az analízis során készített öröklési gráfot. Ez a hatékonyság kárára válhat.
• Az analízis, tervezés és implementáció során használt öröklési gráfok célja más és más, ezeket egymástól függetlenül kell kezelni.

116. Mit jelent az UML asszociáció? Mondjon példákat lehetséges asszociációkra.
Az objektumok és objektum-osztályok más objektumokkal és objektum-osztályokkal lehetnek kapcsolatban.
A UML-ben az általánosított kapcsolatot az asszociációval jelezzük.
Az asszociációkon külön szöveges információ írhatja le az asszociáció jellegét.

117. Mik a konkurens objektumok?
Az objektumok önálló entitások, így alkalmasok párhuzamos implementációra.
Az objektum-kommunikáció üzenet-modelljét közvetlenül lehet implementálni, ha az objektumok egy elosztott rendszerben, különböző processzorokon futnak.

118. Mik a szerverek és az aktív objektumok főbb jellemzői? Az implementáció lehetőségei.
Szerverek: az objektumot párhuzamos folyamatként (szerver) implementáljuk, ahol az objektum operációi belépési pontok lesznek. Ha nincs hívás az objektumra, akkor az felfüggeszti magát és várakozik további szolgáltatás-hívásokra.
Aktív objektumok: az objektumot párhuzamos folyamatként implementáljuk. A belső állapotokat az objektum maga is megváltoztathatja, nem kell hozzá külső hívás.
Implementáció:
Az aktív objektumok attribútumait operációkkal is megváltoztathatjuk, de autonóm módon, belső műveletekkel ezt maguk is megtehetik.

119. Ismertese az objektum-orientált tervezési folyamat fő elemeit!
• **A rendszer kontextusának és felhasználási módozatainak definiálása**.
• **A rendszer-architektúra tervezése**.
• **Az alapvető rendszerobjektumok meghatározása**.
• **A tervezési modellek kidolgozása**.
• **Az objektum interfészek specifikálása**.

120. Mi a rendszer kontextusa és felhasználási módozatai? Hogyan modellezzük ezeket?
A fejlesztendő szoftver és külső környezete közti kapcsolatok feltérképezése.
• **A rendszer kontextusa**: statikus modell, amely leírja a környezetben levő más rendszereket. Alrendszer (subsystem) modellek használhatók más rendszerek jelzésére.
• **A rendszerhasználat modellje**: dinamikus modell, amely a rendszer és környezetének interakcióját mutatja be. Use-case modellek használhatók az interakciók leírására.
Modellek:
Use-case modellek:
• Use-case modellek használatával a rendszer valamennyi interakciója leírandó.
• A use-case modellek a rendszer szolgáltatásait ellipszisek segítségével jelölik. Az interakcióban résztvevő entitást pálcikaember jelzi.

121. Mi az architektúra tervezés? Milyen modelleket használhatunk?
A rendszer és környezete közötti interakciók megértése után ez az információ felhasználható a rendszer-architektúra tervezésére.
Egy architektúra-modell ne tartalmazzon több, mint 7 entitást.

122. Ismertesse az objektumok azonosítására használható módszereket!
• **A rendszer természetes nyelvi leírásán végzett nyelvi elemzés**.
• **Az alkalmazási környezetbeli kézzelfogható dolgok alapján**.
• **Viselkedési megközelítés**: mely objektumok mely viselkedésben vesznek részt.
• **Szcenárió-alapú analízis**: minden szcenárióban azonosítjuk az objektumokat, attribútumokat és a metódusokat.

123. Mi a tervezési modellek feladata? Mit írnak le a statikus és dinamikus modellek? Adjon példát lehetséges tervezési modellekre!
A tervezési modellek az objektumokat, objektum-osztályokat, valamint ezen entitások közötti kapcsolatokat írják le.
Statikus modellek: a rendszer statikus struktúráját írják le objektum-osztályok és relációik segítségével.
Dinamikus modellek: az objektumok közötti dinamikus interakciókat írják le.

Pl.:
• **Alrendszer (sub-system) modellek**: az objektumok koherens alrendszerekre való logikus csoportosítását adják.
• **Szekvencia-diagramok**: az objektumok közötti interakciók sorozatát írják le.
• **Állapotgép-modellek**: az egyes objektumok hogyan változtatják állapotukat eseményekre reagálva.
• **Egyéb modellek**: use-case modellek, aggregációs modellek, általánosítási modellek, stb...

124. Mi az objektum interfészek specifikációjának jelentősége? Milyen módszerek alkalmazhatók interfészek definiálására?
Az objektum interfészek specifikációjának jelentősége: az objektumok és más komponensek párhuzamosan fejleszthetők legyenek.
Módszerek:
Az UML-ben osztály-diagramot használunk az interfészek specifikálására.

125. Miért előnyös az OOT a terv evolúciója szempontjából?
Az objektum-orientált tervezés nagy valószínűséggel leegyszerűsíti a rendszer evolúcióját.