1. A szoftvertechnológia kialakulása, feladata, eszközei:

XXI. század – az informatika százada. Az informatikai problémák egyre növekvő hányada szoftver úton oldható meg.

'40 Neumann (ENIAC,JONIAC) – elektronikus számítógép.

'50 első elektronikus programvezérelt gépek (elektroncsövek) sok számítással járó tudományos és műszaki feladatok megoldása.

'60 lyukkártya, szalag, nyomtató, háttértár. Szellemileg munkaigényes, de mechanikusan ismétlődő ügyviteli feladatokra, gyorsan változó fizikai folyamatokra (űrkutatás, hadászat).

'70 gyártósorok vezérlése, termelés folyamatához kapcsolódó ügyviteli feladatok → komplex termelésirányító rendszerek.

'80 Pc – kisvállalkozások, oktatás, szórakozás, játék.

'90 kommunikáció, multimédia, távmunka.

A hardver követni tudta az alkalmazások igényeit. Az erőforrások nőttek, a méretek csökkentek.

Nagy rendszereknél több ember tudta csak elvégezni a fejlesztést → a létező módszerek alkalmatlanok a feladatok megoldására. (idő, rejtett hibák, költségek)

Megoldás a program = termék ; az előállításához technológia kell.

Tervezési paraméterek:

- van szolgáltatási funkciója
- minősége
- költsége
- határideje

A szoftvertechnológia tárgya a nagyméretű programrendszerek előállítása.

Ezek jellemzői:

- nagy bonyolultság,
- csapatmunkában készül,
- hosszú élettartam.

Fő feladatai:

- a rendszerrel szemben támasztott követelményeket előre pontosan meg kell határozni.
- A program kidolgozásának menetét meg kell tervezni.
- Gondoskodni kell a szükséges erőforrásokról(HW,SW, ember, cash).
- Dokumentálni kell a készítést és az eredményeket.
- Szervezni, irányítani kell a team-et és az erőforrásokat.
- Igazolni kell hogy a kész termék megfelel-e a követelményeknek.
- Karbantartás.

Ezen feladatok összessége a projektmenedzselés, a felelős pedig a projektvezető.

A szoftvertechnológia célkitűzése:

- minőségi programtermék
- határidőre
- meghatározott költségen.

A technológia összetevői:

- Módszerek a programkészítés fázisaira
- szabványok
- programeszközök, programfejlesztési környezet
- irányítási módszerek a programkészítés folyamatának vezérlésére, szervezésére.

2. Projektmenedzselés feladata, eszközei

Szoftverprojekt a szoftvertermék előállításával kapcsolatos tevékenység, az ajánlattevéstől a kész termék átadásáig.

A projektvezető feladata:

- ▷ a projekt tevékenységeinek ütemezése
- ▷ a projekt tevékenységeinek összehangolása
- ▷ felelős a projekt sikeréért.

A projektvezető felelőssége:

- ▷ a termék nyújtsa a megkívánt szolgáltatásokat.
- ▷ a termék minősége feleljen meg az előírt követelményeknek.
- ▷ készüljön el határidőre.
- ▷ az összköltség ne lépje túl a keretet.

A projekt vezetése:

1. ajánlatkészítés
2. a projekt megvalósításának megtervezése
3. költségbecslés
4. szükséges erőforrások biztosítása
5. a munka menetének irányítása, ellenőrzése
6. az eredmények bemutatása, átadása

Ajánlat:

- ▷ Mi a tárgya?
- ▷ Mik a célok?
- ▷ Mi a téma gazdasági, műszaki háttere?
- ▷ Hogy éri el a kitűzött célt
- ▷ Mik az ajánlattevők szakmai hátterei (referencia)
- ▷ A projekt tartalmi leírásai fő fázisai szerinti bontásban, ráfordítással költséggel, határidőkkel
- ▷ Milyen szakmai, piaci előnyöket jelent a projekt sikere?

A projekt megvalósításának megtervezése:
Feladatok:

- ▷ a projekt részfeladatainak meghatározása
- ▷ mérföldkövek kijelölése
- ▷ szükséges idő megbecslése
- ▷ a részfeladatok egymástól való függésének meghatározása (függési táblázat).

3. Programfejlesztési modellek, összehasonlításuk

1. egyszerű programfejlesztési modell
2. programfejlesztés specifikációval
3. nagy rendszerek fejlesztésének általános modellje
4. programfejlesztés modellalkotással

3.1. Egyszerű programfejlesztési modell:

- ▷ egyszemélyes programfejlesztésnél
- ▷ a feladat könnyen áttekinthető, modellezhető, megfogalmazható
- ▷ az eredményt szoftvereszközök állítják elő
- ▷ az eredményt a feladattal vetjük össze, a javítást közvetlenül a programozási nyelvű leírásban hajtjuk végre.
- ▷ kis programok létrehozásának modellje

3.2. Programfejlesztés specifikációval:

- ▷ a programkészítés folyamatát matematikailag is kezelni tudjuk.
- ▷ a feladat megoldása először beszélt nyelven, ennek eredménye egy informális leírás.
- ▷ az informális leírást összevetjük a megoldandó feladattal, ez szintaktikailag és szemantikailag nem egyértelmű.
- ▷ formális leírás(absztrakt program). Ez egyértelmű. Ezt összevetjük az informális leírással.
- ▷ az informális leírás alapján megfogalmazzuk a specifikációs nyelven a megoldás olyan tulajdonságait, amik explicit módon nem szerepelnek a specifikációban.(verifikációs analízis).
- ▷ a konkrét nyelven megírt programot összevetjük az absztrakt programmal(verifikáció).
- ▷ tesztelés után javíthatunk bármilyen lépésnél.

Előnyök:

- ▷ magas szintű, az absztrakt szintaxis és a szemantika szempontjából pontos megoldás a programkészítés korai fázisában.
- ▷ az absztrakt leírás formális elemezhetősége.
- ▷ a konkrét leírás helyességének bizonyíthatósága az absztrakt leírás szerint.

Hátrányok:

- ▷ - magas szintű matematikai ismereteket igényel.
- ▷ - nagyobb feladatok leírására alkalmatlan. (áttekinthetetlen) .
- ▷ - a matematikai bizonyítás egyszerű, de a tétel megfogalmazása már nehéz.

3.3. Nagy rendszerek:

- ▷ a specifikációs modell formális leírásának helyébe a programrendszer tervének elkészítése lép. Ez jól struktúrált, többszintű leírás, amely tartalmazza az implementáció számára a szükséges leírásokat, ajánlásokat.

4. Nagy rendszerek fejlesztési fázisainak kapcsolata

A programfejlesztés fázisai:

- ▷ követelmények leírása (1)
- ▷ specifikáció (2)
- ▷ tervezés(vázlatos(3), finom(4))
- ▷ implementáció(5), integráció(6)
- ▷ verifikáció(7), validáció(8)
- ▷ rendszerkövetés(9), karbantartás(10)

A kapcsolatok leírása, kialakult modellek:

1. vízesés modell
2. evolúciós modell
3. Boehm-féle spirális modell

4.1. Vízesés modell:

Hátrányai:

- ▷ a projekt munkájának megszervezése nehéz
- ▷ új szolgáltatások bevezetése minden fázisnál módosítást kíván
- ▷ a validáció az egész életciklus megismétlését követeli meg.

4.2. Evolúciós modell:

- ▷ a megoldás közelítőverzióinak sorozatát állítjuk elő
- ▷ nagyon sok részlet csak a felhasználó fejében létezik, így nem készíthető teljes specifikáció
- ▷ elkészítjük a megoldás első verzióját és ehhez, egy a felhasználói felületet szimuláló prototípust, egyeztetjük a felhasználóval, és ez alapján döntünk a következő verzióról.

Előnyei:

- ▷ működés közben tanulmányozható prototípusok

Hátrányok:

- ▷ csak rövid idő alatt megvalósítható rendszereknél működik.
- ▷ nehézkes az utólagos módosítás.
- ▷ nehéz a projektet áttekinteni.
- ▷ nehezen dokumentálható.

4.3. Boehm-fél spirális modell:

- ▷ a programok iterációkon keresztül nyerik el végső formájukat.
- ▷ egy spirál 4 szakasszal
  1. elérendő célok, alternatívák meghatározása, feltételek definiálása.
  2. az alternatívák kiértékelése, kockázatok kezelése/csökkentése prototípusok készítése.
  3. a fázis termékeinek megvalósítása, validáció.
  4. a következő iterációs lépés megtervezése.

Előnyök:

- ▷ jó dokumentálhatóság
- ▷ a projekt struktúráltsága, az iterációs fázisok szabadon választhatók
- ▷ a fázison belüli szakaszokat a projektvezető határozhatja meg.
- ▷ A fejlesztést, az eredményt mindig a validáció követi.
- ▷ Ciklusonként döntés születhet a következő fázisról.

Problémák:

- ▷ munkaigényes, bonyolult, a szakaszokra bontás költséges és nehezen megoldható.
- ▷ nehéz a munkaerőt párhuzamosan foglalkoztatni – gazdaságtalan.

5. Nagy rendszerek fejlesztési fázisai I.

Megvalósíthatósági tanulmány:

Probléma→megvalósíthatósági elemzés→ megvalósíthatósági tanulmány

- ▷ milyen erőforrások kellenek? (HW, SW környezet, szakemberek)
- ▷ mennyibe kerül
- ▷ mikor lesz kész
- ▷ az üzemeltetés milyen feladatokkal jár, mennyibe kerül?

Követelmények leírása:

- ▷ mi a probléma?
- ▷ mik a korlátozó tényezők?
- ▷ mi a megoldás? (minőség)

Fajtái:

- ▷ funkcionális (szolgáltatások)
- ▷ nem funkcionális (korlátozások az erőforrásokra, környezetre)

Funkcionális követelmények leírása:

- ▷ mi a formája a szolgáltatás kezdeményezésének
- ▷ mik a szolgáltatás bemenő adatai, hogy kell ezeket megadni?
- ▷ mik a szolgáltatások igénybevételének előfeltételei, mik a korlátozások?
- ▷ mi a szolg. kezdeményezésére a válasz, mik az eredmények?
- ▷ mi a válasz formája, hol jelenik meg?
- ▷ mi a bemenő adatok és az eredmények közötti reláció?

A funkcionális követelmények leírásának módszerei:

- ▷ beszélt nyelven történő leírás
- ▷ formáknak alávetett leírás
- ▷ követelmények leírására szolgáló nyelvek, grafikus modellezési eszközök (pl.:PDL, PSL, SADT). Ezek programszerű nyelvek, melyek absztrakt leírást tesznek lehetővé
- ▷ formális matematikái leírások (egyértelmű szintaxis és szemantika)
  - előfeltételes forma
  - axiomatikus specifikáció

Követelmények elemzése:

- ▷ jó-e a követelmények leírása?
  - nincs benne ellentmondás? (konzisztencia vizsgálat)
  - teljes-e a leírás? (komplettség vizsgálat)
- ▷ a követelmények leírása megfelel a problémának? (validáció vizsgálat)
- ▷ megvalósítható? (megvalósíthatósági vizsgálat)
- ▷ tesztelhető? (tesztelhetőség vizsgálata)
- ▷ módosítható, továbbfejleszthető? (nyíltság vizsgálata)

A prototípus magas szintű programozási nyelven megírt, a külső viselkedés szempontjából helyes megoldása a problémának:

- ▷ konzisztencia, komplettség, validáció vizsgálat.
- ▷ félreértések tisztázása a felhasználók és a fejlesztők között.
- ▷ a felhasználó megítélheti az alkalmazás hasznosságát.
- ▷ a felhasználók felkészítésének eszköze lehet
- ▷ elősegítheti a követelmények specifikálását
- ▷ Szerepe a követelmények meghatározásában

6. Nagy rendszerek fejlesztési fázisai II.

Programspecifikáció:

- ▷ mik a bemenő adatok?
- ▷ mik az eredmények?
- ▷ mi a reláció a bemenő adatok és az eredmény adatok között.

A specifikációs módszerek csoportosítása:

- ▷ informális (beszélt nyelven történő leírás)
- ▷ formális
- ▷ formáknak alávetett (a kettő között helyezkedik el, egyértelműen definiált formalizmus + beszélt nyelv bizonyos helyeken)

Tervezés:

1. a statikus modell megalkotása
   - ▷ mi a rendszer szerkezete?
   - ▷ milyen programegységekből épül fel?
   - ▷ mi az egységek feladata?
   - ▷ mi az egységek közötti kapcsolat?
2. dinamikus modell
   - ▷ hogyan oldja meg a rendszer a problémát
   - ▷ milyen egységek működnek együtt a megoldásnál
   - ▷ mik az üzenetek az együttműködő egységek között
   - ▷ mik a rendszer, az egységek állapotai
   - ▷ milyen eredmények hatására valósulnak meg az állapotok közti átmenetek?
3. funkcionális modell
   - ▷ milyen adatáramlás kell a szolgáltatásokhoz?
   - ▷ milyen leképezések kellenek az adatáramlásokhoz?
   - ▷ mik az ajánlások az implementáció számára? (implementációs stratégiára, programozási nyelvre, és a tesztelési stratégiára vonatkozó ajánlás)

Tervezés fajtái:

- ▷ procedurális (funkcionális) elvű
- ▷ objektum elvű

6.1. Procedurális tervezés:

- ▷ a megvalósítandó funkciókból, műveletekből indulunk ki, ez alapján bontjuk fel a rendszert kisebb modulokra, majd ezeket finomítjuk az egyes szinteken (felülről lefelé)
- ▷ ellentéte az alulról felfelé
- ▷ tipikus példája a strukturális dekompozíció

A strukturált tervezés eredményei:

- ▷ a program struktúradiagramja
- ▷ input-output tábla
- ▷ a modulok definíciója
- ▷ az adatszótár

6.2. Objektumelvű tervezés:

- ▷ a funkciók helyett az adatokat állítjuk a középpontba
- ▷ az adatok felelnek meg az objektumoknak
- ▷ az adatokat csoportosítjuk viselkedésük szerint
- ▷ a kialakult osztályok, illetve egyedek közötti kapcsolatokat vizsgáljuk, azok állapotait azonosítjuk.

A terv minőségének mutatói:

Kohézió az egy programon belüli komponensek összetartozása.

- ▷ laza kohézió:
  - véletlenszerű
  - időbeli
  - logikai
- ▷ szoros kohézió:
  - kommunikációs
  - funkcionális

Összekapcsolódás (coupling): a programegységek mennyire függenek egymástól

Módosíthatóság a programegységek legyenek önállóak, függetlenek a rendszer más komponenseitől

Fejleszthetőség

Bonyolultság a rendszer áttekinthetősége, megérthetősége.

7. Nagy rendszerek fejlesztési fázisai III.

Implementáció

- ▷ hogyan ábrázoljuk az adatokat? (reprezentáció)
- ▷ hogyan valósítjuk meg a leképezéseket, eseményeket? (algoritmusok választása, optimalizálás)

Stratégiák

- ▷ alulról-felfelé (bottom-up)
- ▷ felülről-lefelé (top-down)
- ▷ modul, alrendszer izoláció
- ▷ objektumelvű modell alapján automatikus kódgenerálás

7.1. Alulról-felfelé

Előny könnyű az elkészült részeket tesztelni

Hátrány a szerkezeti hibák későn derülnek ki

7.2. Felülről-lefelé

Előny a szerkezeti hibák hamar kiderülnek

Hátrány csak a teljes programot lehet tesztelni

7.3. Modul, alrendszer izoláció

Már kipróbált eljárásokat, modulokat használunk fel

7.4. Objektumelvű

Az előző továbbfejlesztése öröklődéssel, aggregációval, kompozícióval

A jó programozási eszközrendszer, szoftverfejlesztési környezet jellemzői:

- ▷ az eszközök egységes rendszert alkotnak
- ▷ az eszközrendszer konstruktív
- ▷ tartalmaz rendszerkonstrukciót támogató eszközöket
- ▷ támogatja a team-munkát

A jó implementációs stílus jellemzői:

- ▷ az absztrakció különböző szintjeinek alkalmazása
- ▷ az öröklődés használata
- ▷ absztrakciós szintekre bontás osztályon belül(deklaráció+megvalósítás)
- ▷ korlátolt láthatóság
- ▷ információ elrejtés (information hiding)
- ▷ információ beburkolás (encapsulation)

Verifikáció a specifikáció szerinti helyesség igazolása

Validáció annak ellenőrzése, hogy a rendszer teljesíti-e az előírt minőségi tulajdonságokat

Tesztelés az ellenőrzés folyamata.

Feladata a rendszer hibáinak működési rendellenességeinek kimutatása, lokalizálása.

Hiba a rendszer nem a specifikáció szerint működik (verifikáció).

Működési rendellenesség ha nem felel meg valamilyen előzetes követelményeknek (validáció)

A tesztelés két szakasza

Egységteszt a rendszer elemeinek moduljainak független tesztelése

Rendszerteszt a teljes rendszer tesztje, a modulok miként működnek együtt.

A tesztelés módjai

Fekete doboz a tesztelendő programrészt ismeretlennek tekintjük

Fehér doboz ismert programrész.

8. Nagy rendszerek fejlesztési fázisai IV.

Ebbe a fázisba a kész rendszerek, kapcsolatok, feladatok tartoznak. Ezek:

Karbantartás az üzembe állítás után szükséges szoftver jellegű munkák

Rendszerkövetés a felhasználókkal való kapcsolattartás menedzsment jellegű dokumentációs feladatai.

A karbantartási munkák csoportosítása:

- ▷ a használat során felmerülő rejtett hibák kijavítása
- ▷ a rendszer új üzemeltetési környezetbe helyezése (adaptáció). Ez lehet:
  - új HW
  - új SW környezet
- ▷ az új igényeknek megfelelő módosítások (továbbfejlesztés)
  - új szolgáltatások
  - kiterjesztés új adatbázisokra
  - kiterjesztés hálózati szolgátatásokra
  - minőségi mutatók javítása

Költségek:

- ▷ rejtett hibák javítása : 10%
- ▷ adaptációs munkák: 25%
- ▷ továbbfejlesztés: 65%

A rendszerkövetési munkák csoportosítása:

- ▷ konfigurációk nyilvántartási feladatai (az alapkonfiguráció és annak szabványos kiterjesztésének meghatározásai)
- ▷ verziók menedzselése(hibajavítások és fejlesztések nyilvántartási rendszere)
- ▷ dokumentáció menedzselése (elektronikus és nyomtatott)
- ▷ rendszer generálása (rendszer összeállítása komponensekből)

Dokumentáció:

- ▷ mire használható a szoftver?
- ▷ milyen környezetben használható?
- ▷ milyen korlátozások között futtatható?
- ▷ mi a program előélete?
- ▷ hogyan installálható?
- ▷ hogyan használható?
- ▷ kik készítették?
- ▷ milyen modulokból épül fel?
- ▷ milyen osztályok fordulnak elő és mi ezek kapcsolata?
- ▷ milyen a rendszer dinamikus viselkedése?
- ▷ az osztályok miként lettek implementálva?
- ▷ teszteléssel kapcsolatos információk.

9. Az objektumelvű programozás kialakulása:

- ▷ a procedurális rendszer hibáinak kiküszöbölésére fejlesztették ki.
- ▷ bevezették az absztrakt adattípust és azt kiegészítve a típusöröklődést objektumelvű programozás = adatabsztrakció + absztrakt adattípus + típusöröklődés

A támogatott típusok a következő BNF-fel:
<típus>::=<egyszerű> | <összetett>;
<egyszerű>::=integer | real | boolean|..;
<összetett>::=vector | array | record | ... | <absztrakt adattípus> | <öröklődéssel származtatott típus>.

Absztrakció A programozás adott szintjén a megoldás szempontjából lényegtelen részek elhanyagolása

Két fő formája:

1. **Paraméteres absztrakció**: a formális paraméterekhez rendelhető aktuális paraméterek konkrét tulajdonságainak elhanyagolása.
2. **Specifikáció szerinti absztrakció**: az adat reprezentációjának és a rajta értelmezett művelet implementálásának elhanyagolása. Ezen belül:

   Applikatív procedurális absztrakció az absztrakt rendszert használjuk fel az adat ábrázolására és ebben a rendszerben írjuk fel az adathoz tartozó műveletek algoritmusait.

   Külső felület szerinti absztrakció a műveletek algoritmusait nem adjuk meg, csak a bemenő adatok halmazát és az egymáshoz rendelt (input-output) párok halmazát.

   Axiomatikus absztrakció nincs ábrázolás, nincs megvalósítás, a program jelentését axiómákkal adjuk meg.

Egyszerű adattípus az (A,F) pár, ahol A az adatok halmaza, F pedig a műveletek véges halmaza.

Összetett adattípus az (A,F) rendezett pár, ahol A={A0,A1,...,An} és A0 a típusobjektumok halmaza, A1,...,An paraméterhalmazok és F = {f0,...,fn} a műveletek véges halmaza.

Típusosztály Olyan négyes (PAR, EXP, IMP, BODY) ahol
PAR=<a paraméterek tulajdonságai>
EXP=<a típusobjektumok halmaza és a rajta értelmezett műveletek>
IMP=<a típusobjektumok ábrázolásához más osztályból átvett szolgáltatások leírása>
BODY=<a típusosztály ábrázolása, megvalósítása>

Típusöröklődés specializációval

1. a subclass átveszi a superclass összes absztrakt tulajdonságát
2. a típushalmaz, a paraméterhalmaz és a műveletek nevei átdefiniálódhatnak, de a jelentésük nem változhat.
3. a subclass típushalmazának a superclass típushalmaza felel meg.
4. a subclass leírása:
   - ▷ új szolgáltatásokkal bővülhet
   - ▷ új paraméterek definiálhatók
   - ▷ az input része módosulhat
   - ▷ új ábrázolási forma és új megvalósítás léphet fel

A specializáció következményei:

- ▷ polimorfizmus
- ▷ dinamikus összekapcsolás

Polimorfizmus a specializáció miatt minden változónak két típusa van:

- ▷ statikus típus, amelyet a deklarálással kap
- ▷ dinamikus típus ami deklaráláskor megegyezik a statikussal, de értékadásnál megváltozhat.

Dinamikus összekapcsolás a dinamikus típusnak megfelelő kiszámítási szabály hozzárendelése a függvényhez, attribútumhoz a végrehajtás pillanatában.

Típusöröklődés specializációval

- ▷ A subclass átveszi a superclass összes absztrakt tulajdonságát
- ▷ A tulajdonságok átvételekor a típushalmaz, a paraméterhalmazok és műveletek nevei újradefiniálódhatnak.
- ▷ A subclass típushalmazának a superclass típushalmaza felel meg
- ▷ Az átvett műveletek jelentése nem változhat meg.

10. Az objektumelvű modellezés alapjai

Nézetrendszerek

az ember vizuális lény, problémamegoldást valamilyen szempontok alapján szemléltetni tudta.

Használati szempont Szolgáltatás nyújtása (személy, rendszer, programok)

Szerkezeti, strukturális, statikus szempont rendszer egységei, feladatuk, kapcsolatuk egymással.

Dinamikus szempont rendszer részegységeinek viselkedése

Implementációs szempont Szoftverkomponensek, kapcsolatuk egymással.

Környezeti szempont hardver és szoftver erőforrások

UML

(gyakorlati diagramokat szabályos, egységes nyelvben foglalja össze).

A nyelv szintaxissal, szemantikával rendelkezik → szoftvereszközökkel objektumelvű programozási nyelvre transzformálható

UML diagramjai (kapcsolódásuk nézetrendszerhez)

statikus szempont szerint osztálydiagram(class), objektumdiagram(object)

dinamikus szempont szerint állapotdiagram(statechart), szekvenciadiagram(sequnce), együttműködési diagram(collaboration), aktivizációs diagram(activity)

implementációs szempont szerint Komponensdiagram (component), alrendszerdiagram

környezeti szempontok szerint konfigurációs diagram(deployment): hardver-szoftver konfigjainak szemléltetése.

használati szempont használati esetek diag-ja(use case): rendszer-felhasználó kapcs. leírása.

Statikus modell = osztálydiagram + osztályleírások, objektumdiagram + objektum leírások.

Objektum:

1. azonosítható, egymástól különbözőek
2. attribútumok tartoznak hozzá, melyek között formális paraméterek is lehetnek.
3. állapota van, ezek az attribútumok konkrét értékei
4. műveletek tartoznak hozzá
5. korlátolt láthatóság

   látható rész interfész, export-import műveletek

   láthatatlan rész objektum ábrázolása, műveletek megvalósítása
6. megjelenési formák: absztrakt forma, konkrét forma
7. az osztály egy példánya („is-a” reláció – objektum – osztály relációja)
8. szabványos felület, hozzáférések .

Objektum = identitás + megnyilvánulás + állapot.
Azonosítás történhet:

- ▷ névvel,
- ▷ állításra adott válasszal, ami csak az adott objektumra igaz.

Művelet:

- ▷ export műveletek: (más objektumok végzik rajta pl: verem(push, pop, top))
- ▷ import műveletek: (ezeket igényli, hogy az export szolgáltatásokat nyújtani tudja. Ezeket más objektumokon végzi el.)

Export:

- ▷ konstruktor műveletek (létrehozás, felépítés)
- ▷ kiértékelés (objektum jellemzőire kérdez rá)
- ▷ szelektor (objektum bizonyos részét kiemelik)
- ▷ állapot megváltoztató művelet (attribútum értékének változtatása)
- ▷ iterátor eljárás (objektum felépítésében részt vevő komponensek bejárása).

Műveletek típusai:

kliens aktív objektum (csak másik objektumon végez műveleteket, de rajta senki. Nincs export felülete.).

szerver passzív objektum (csak export felülete van).

ágens általános objektum modell: van import, export felülete.

11. Osztály, diagramok, jelölések

Az osztály jellemzői:

- ▷ hasonló tulajdonságú objektumok egy halmaza
- ▷ van neve, amelyet az osztályba tartozó összes objektum örököl
- ▷ lehetnek attribútumai, paraméterei, amelyek az objektumoknak is közös építőkövei.
- ▷ tartoznak hozzá szolgáltatások, operációk, műveletek (export felület), amelyek lehetnek objektum szintűek, vagy az osztály egészére vonatkozóak.
- ▷ tartozhat hozzá import felület, amely az általa igényelt szolgáltatások definícióját tartalmazza.
- ▷ rendelkezhet megvalósítási résszel, amely a következőkből áll:
  - paraméterek leírása
  - szolgáltatások leírása
  - import felület leírása
  - megvalósítás leírása
- ▷ van látható és láthatatlan része
- ▷ lehet absztrakt (nincs import része és megvalósítási része)
- ▷ lehet konkrét (minden szolgáltatáshoz definiált annak megvalósítása)
- ▷ 3-féle hozzáférési módja lehet:
  - public;
  - protected;
  - private
- ▷ lehet paraméteres osztály (sablon), aminek a definícióit olyan formális paraméterekkel adjuk meg, amelyeknek sem típusa, sem korlátozása nincs meghatározva.

Sablon nem lehet példány vagy általánosítás (superclass).

Sablon lehet specializáció (subclass), tartalmazhat saját formális paramétereket a definiáláshoz.

Az osztály leírásának összetevői:

paraméter rész leírása az osztály objektumainak felépítésének jellemzői (attribútum), fajtái: objektum attribútumok, osztályattribútumok

szolgáltatások leírása objektum operációk, osztály operációk, korlátozások

import rész leírása a más osztályokból igénybevett szolgáltatások leírása.

megvalósítás leírása szerkezeti tulajdonságainak ábrázolása és viselkedésbeli tulajdonságainak konkrét implementációja.

Osztálydiagram

A problématérben a megoldás szerkezetét leíró, összefüggő gráf, amelynek csomópontjaihoz az osztályokat, éleihez pedig az osztályok közötti relációkat rendeljük. Az osztályok közötti reláció lehet: öröklődés, asszociáció, aggregáció, kompozíció. (sign)

Objektumdiagram

Az objektumdiagram egyszeresen összefüggő gráf, amelynek csomópontjaihoz az objektumokat, éleihez az objektumok közötti relációt rendeljük. Egy osztály diagramhoz több objektumdiagram tartozhat.

Annotáció

az UML nyelv szemantikai kiegészítése.

12. Objektumosztályok közötti kapcsolatok I.

Asszociáció

két osztály közötti absztrakt reláció, amely kétirányú társítást fejez ki. Azért absztrakt, mert a reláció konkretizálása osztályok objektumainak összekapcsolásában valósul meg (konkrét esetben összekapcsolás, absztrakt esetben társítás).

- ▷ lehet reflexív (azonos osztályon belüli objektumok összekapcsolását is engedi).
- ▷ társulhat hozzá annak neve, azonosítója.
- ▷ irány is társulhat hozzá, amely az aktív objektumtól a passzív felé mutat.
- ▷ részleteinek leírása a hozzá társult osztályban kaphat helyet.
- ▷ az összekapcsolt objektumoknak lehet multiplicitása:
  - pontosan i : i
  - i és j között: i. . . j
  - valamennyi: *
  - legalább i: i..*
- ▷ az asszociációban részt vevő objektumoknak lehet szerepe:
  - névvel azonosított szerep
  - kiemelt szerep
  - sorrendiségi szerep
- ▷ minősítő társulhat hozzá, amelynek értékei az objektumokat a társítás szempontjából diszjunkt partíciókhoz rendelik.

13. Objektumosztályok közötti kapcsolatok II.

Aggregáció

olyan kapcsolatot fejez ki, mint: egész és annak részei, vagy felépítmény és annak komponensei.

- ▷ speciális asszociáció
- ▷ azt fejezi ki, hogy az egyik osztály objektumai részét képezik egy másik osztály objektumainak.
- ▷ tranzitív
- ▷ aszimmetrikus
- ▷ lehet reflexív
- ▷ ha (A is an aggregation of B) és f az A osztály egy szolgáltatása, akkor B osztálynak is szolgáltatása lesz.
- ▷ ha attr az A osztály egyik attribútuma, akkor B-é is lesz.
- ▷ ha A és B osztályok aggregációban vannak, akkor A és B objektumai egymástól függetlenül is létezhetnek.
- ▷ különböző aggregátumoknak lehetnek közös komponensei.

Kompozíció

a legerősebb kapcsolat

- ▷ speciális aggregáció
- ▷ az egyik osztály objektumai fizikailag tartalmazzák a másikat.
- ▷ a kompozíciós kapcsolat és az attribútum jellegű kapcsolat két objektum között szemantikailag azonos, csupán a grafikai megjelenésük különbözik. Az attribútum jelölése a kompozíció jelölésének egy egyszerűsített formája.
- ▷ egy komponens objektum legfeljebb egy aggergációs objektumhoz tartozhat.
- ▷ az aggregációnak tetszőleges számú kompozíciója lehet.
- ▷ az aggregációs objektum és annak komponensei azonos életciklusban léteznek, egyszerre jönnek létre és szűnnek meg.

Öröklődés

létrehozzuk az általános tulajdonsággal rendelkező osztályt, majd ennek tulajdonságait átvéve származtatjuk a speciálisabb tulajdonsággal rendelkező osztályokat.

- ▷ egy általános konstrukció és egy speciális konstrukció közti kapcsolat.
- ▷ a speciális osztály az általánosból származtatással jön létre.
- ▷ a speciális osztály átveszi az általános tulajdonságait, új jellemzőket vezethet be és az átvett jellemzőket újrafogalmazhatja.
- ▷ a származtatás az öröklődési technika felhasználásával történik.

Speciális esetei:

- ▷ specifikációs öröklődés(absztrakt tulajdonságokat vesz át)
- ▷ implementációs öröklődés (konkrét és absztrakt tulajdonságokat vesz át)
- ▷ a származtatás nem szimmetrikus
- ▷ a származtatás nem lehet reflexív
- ▷ az általánosítás és a specializáció is lehet többszörös
- ▷ a származtatás „is a kind of” reláció.

14. Állapotdiagram

Állapot az áobjektum állapotát az attribútumok konkrét értékeinek n-esével jellemezzük

1. van azonosítója.
2. általában esemény(sorozat) hatására jön létre. (*pre-events*)
3. Az állapot időben mindaddig fennmarad, amíg az objektumok attribútumainak értékei kielégítik az állapothoz rendelt invariánsokat.
4. gyakran azonosítóként használjuk a belső műveltek nevét
5. Az állapot megszűnése egy esemény hatására következik be (*post-events*)
6. állapot lehet részállapotok általánosítása (ekkor van állapotdiagramm is)
7. Állapot lehet *pszeudoállapot:*
   (a) kezdéskor a külső állapot
   (b) befejezéskor a külső állapot
   (c) hisztorizációs állapot, melyhez hisztorizációs indikátor társul. (lehetőséget biztosít, hogy adott pillanatban felfüggesszük az állapotokat, majd bizonyos idő után újra ugyanonnan folytatódjon az állapotok változása)

Esemény eseménynek nevezzük azt a tevékenységet, történést, amely valamely objektum állapotát megváltoztatja.

1. lehet paraméteres vagy paraméter nélküli
2. események között sorrendiség állhat fent (megelőző, rákövetkező)
3. eseménynek lehet előfeltétele

Állapotdiagram összefüggő irányított gráf, csomópontjaihoz rendeljük az állapotokat, éleihez eseményeket rendelünk. (2 csúcsot több irányított él is összeketöthet).

Esemény és akció: (egy időpillanat alatt játszódik le).

Akció időpillanathoz tartozik, esemény időben elcsúszhat

Esemény fázisai

entry fázis belépés akciója, ez indítja el az eseménysorozatot, ami eredménye a hozzá rendelt állapot.

event fázis adott állapothoz kötődik.

exit fázis esemény befejezése, hozzá rendelt állapotból való kilépés.

Bonyolultság n állapot esetén állapotátmenetek száma $n\,(n-1)$.

Cél egyszerűbbé és átláthatóvá tenni az állapotdiagramot.

Általános módszer

Állapotok általánosítása

1. véges számú részállapot összessége
2. részállapotok örökölhetik az általánosított állapot tul.-t (attribútumait, eseményeit, akcióit)
3. részállapotok lehetnek általánosított állapotok is.
4. az általánosított állapotban az objektum mindig valamilyen részállapotban van.
5. állapotdiagram a részállapotok közti átmenetet tüntetik fel.
6. olyan állapotátmenetek tartoznak az állapotdiagrambeli részállapotokhoz, amik az általánosított állapotot nem váltpztatja meg.
7. van legalább egy olyan részállapot, ami az *entry* akcióját örökli kezdő állapotból a megfelelő részállapotba vezet el.
8. *entry* akció hatására létrejön a megfelelő objektum, és az *exit* akció hatására megsemmisül.

Állapotok aggregációja

1. általa létrejövő állapot egymástól független részállapotok véges halmaza.
2. Minden részállapothoz állapotdiagram tartozik.
3. részállapot lehet általánosított állapot is.
4. minden részállapothoz tartozó állapotdiagramban kell lennie legalább egy olyan állapotnak, amibe a részállapot az *entry* akció miatt kerül.
5. ha az aggregátum rendelkezik *exit*tel, akkor legalább egy olyan állapot lesz a részállapotok állapotdiagramjában, ami az aggregátum *exit*-jét örökli.
6. aggregációs állapot objektuma az aggregációt alkotó részállapotban egyidejűleg létezik.
7. az állapotok aggregációja az állapotban belüli állapotdiagramok közötti párhuzamosság egy megjelenési formája.

15. Szekvenciadiagram

Rendszer működésének leírása; objektumok egymásnak üzenetetküldenek(időbeni sorrend)

Alapfogalmak

- ▷ osztályszerep
- ▷ osztályszerep életvonala
- ▷ aktivizációs életvonal
- ▷ üzenet.

Osztályszerep <szerep neve> : <osztály neve>

Életvonal osztályszerep időben való létezése.

Aktivizációs életvonal művelet végrehajtás közben más objektumok vezérli őket

- ▷ objektum létrehozása, megsemmisítése: objektum létrejön egy másik objektum által küldött üzenet hatására, és meg is semmisíthető.
- ▷ objektum aktivizációja
- ▷ speciális eset: rekurzív aktivizáció
- ▷ centrálisan vezérelt
- ▷ decentrálisan vezérelt

Üzenetek

egyszerű szöveg közönséges eljáráshívás

szinkronizációs üzenet küldő elküldi az üzenetet → küldő blokkolt állapotba kerül, amíg fogadó nem fogadja az üzenetet.

időhöz kötött várakozás küldő legfeljebb a megjelölt ideig várakozik, utána tovább folytatja a tevékenyégét

randevú fogadó várakozik, hogy a küldő üzenetet küldjön neki.

aszinkron küldő folyamat nem szakad meg

visszatérési üzenet az üzenetet egy aktivizált objektum küldi az aktivizáló objektumnak akkor, amikor befejezi a tevékenységét, és a vezérlést visszaadja.

Szekvencia kiegészítései (Message sequence chart)

Hierarchia ábrázolása

dekompozíció objektumát, ill osztályszerepét egy másik szekvenciadiagramban fejtjük ki. Az üzenetk sorrendje nem vátozhat.

hivatkozás a szekvenciadiagramm egy időintervallumnak megfelelő részét kiemeljük külön diagramba. (lekerekített sarkú téglalap)

Választások ábrázolása:

- ▷ pl egy üzenetre csak egy bizonyos feltétel mellett kerül sor. [opt]
- ▷ egy kifejezés értékétől függően eltérő részek lehetnek [alt]
- ▷ iteráció ábrázolása [loop]
- ▷ párhuzamosság ábrázolása egymással egy időben zajló részben. Az egyes részeken belül az üzenetek sorrendje rögzített. [par]

16. Együttműködési, aktivációs diagram

Funkcionális modell: a feladat megoldása érdekében egymás után végrehajtandó tevékenységek folyamatát írja le, a tevékenyéskeg, leképezések között áramló adatokkal együtt.
olyan dinamikus modell, ami az adatok mozgására válaszol.

Együttműködési diagram (kollaborációs):

- ▷ Osztályok, objektumainak együttműködése, üzenetek cseréje. (objektumdiagramban asszociációs kapcsolatok kötik össze). Objektumdiagram kiterjesztése.
- ▷ Üzenetnek lehet argumentuma, eredménye.
- ▷ Üzenetnek egy osztályon belül lehet több címzettje is.
- ▷ Részt vehet rendszeren kívüli objektum (aktor) is. Első üzenetet ő indítja.

Aktivációs diagram

probléma megoldásának lépéseit szemléltető, párhuzamosan zajló vezérlési folyamat.

Üzleti életben előforduló szervezetek munkafolyamatainak modellje.

- ▷ életfolyam: objektumoknak életfolyamokat feleltetünk meg, időben felülről lefelé.
- ▷ sávos alapú: objektumnak függőleges sáv felel meg. Tevékenység sorrendjét nyilak adják meg.

Elemei tevékenységek, folyamok <tevékenység neve>

17. Adatfolyam diagram

- adattár (data store) : objektumai szerverek
- terminátor: objektumai aktorok v. ágensek
- folyamat és objektumfolyam

Általában szintekre tagolt

Végrehajtási gráf

Olyan eszköz amivel a létrehozandó rendszer teljesítményét olvashatjuk le (nem része az UML-nek de könnyen létrehozható a diagramok felhasználásával.).

irányított gráf, csúcsai a rendszer tevékenységei, élei a sorrendet ábrázolják.

Egyszerű csúcs: adott szinten tovább nem bontható

Ismétlési csúcs: ez utáni csúcsok n-szer ismétlődnek, a ciklus utolsó csúcsából ebbe a csúcsba mutat él.

Választási csúcs: az ehhez csatolt csúcsok végrhajtása feltételes, minden esethez tartozik egy végrehajtás valószínűség.

Párhuzamossági csúcs: ehhez csatolt csúcsok tevékenységei párhuzamosan hajtódnak végre, úgy hogy mindegyik befejeződik, mielőtt ebből a csúcsból kilépnénk.

Hasító csúcs: csatolt csúcsok párhuzamosan futó új szálakat reprezentálnak, amelyek befejeződése előtt is továbbléphetünk.

Kiterjesztett csúcs: olyan lépés, amit más végrehajtási gráfban fejtünk ki.

Szinkronizációs csúcsok:

1. hívó folyamatra vonatkozó

   szinkronhívás a hívó válaszra vár

   aszinkronhívás nincs válasz nem kell várni

   késleltetett szinkronhívás a hívó válaszra vár közben azonban a feldolgozás zajlik.

2. Hívott folyamatra vonatkozó. Csak azt kell jelölnünk, hoyg van-e válasz.

18. Alrendszerdiagram

Alrendszer (subsystem): egységként kezelt szoftverkomponensek rendszere.
Pl.: csomag(package) v. fájlrendszer egy operációs rendszerben.

1. alrendszer önálló névvel ellátott programegység.
2. logikai szempontok alapján szoftverkomponensek egységként kezelt rendszere.
3. az alrendszert modulok ill. más alrendszerek alkotják.
4. önállóan oldja meg feladatát.
5. osztályuknak objektumai az alrendszerek között definiált szabványos kapcsolaton keresztül eléri más alrendszerek szolgáltatásait.

Rendszerszerkezetek:

ügyfélszolgáltatókliens-szerver kapcsolat

ügyfél szerepét eljátszó objektum

- ▷ ismeri a szolgáltatót
- ▷ ismeri a szolgáltatás igénybevételének formáját
- ▷ tudja hogy mit nyújt a szolgáltatás
- ▷ szolgáltatás teljesítésének részletei rejtve maradnak számára

Szolgáltató szerepét eljátszó objektum:

- ▷ nem ismeri az ügyfelet
- ▷ nem tudja hogy az ügyfél milyen célból veszi igénybe
- ▷ ismeri a megvalósítás módját

Egyenrangú partner (peer to peer) kapcsolat:

- ▷ tudnia kell a szükséges szolgáltatásról hogy melyik osztály objektuma biztosítja neki.
- ▷ ismeri a szolgáltatás igénylésének formáját.
- ▷ tudja hogy mi a szolgáltatás.

Rendszerek osztályozása

- ▷ rétegszerkezetű architektúra
- ▷ particionális szerkeztű
- ▷ vegyes szerkezetű

Rétegszerkezetű architektúra

1. Az alrendszerek egymásra épülő virtuális világot alkotnak.
2. Minden virtuális világ egy réteg.
3. A vitruális világ az alatta elhelyezkedő világokat ismeri, azoknak a szolgáltatásait igénybe tudja venni. Objektumai ekkor kliensként viselkednek.
4. A fölötte lévő rétegekről nem tud semmit, objektumai ekkor szerverként viselkednek.

Zárt rétegszerkezet esetén a magasabb szinteken lévő rétegek objekutmai csak a közvetlenül alattuk lévő objektumok szokgáltatásihoz férnek hozzá.

Nyílt rétegszerkezet esetén egy adott szintű réteg objektumai több alacsonyab szinten elhelyezkedő objektumhoz is hozzáférnek.

Particionális szerkezetű architektúra

1. A kapcsolat szempontjából egymással egyenrangú alrendszerekből álló szerkezet.
2. Az egyenrangúság azt jelenti, hogy bármely alrendszer objektuma egyenrangú pertnerkapcsolatban állhat bármelyik másik alrendszer objektumával, azaz egymás szoláltatásait kölcsönösen igénybevehetik.

Vegyes szerkezetű architektúra

1. Rétegszerkezetekből és particionális szerkezetkből áll.
2. A vertikális a rétegszerkezet.
3. A horizontális a paricionális szerkezet.

19. Komponens, környezeti diagram

UML diagramok amik a rendszer elkészítésénél lévő implementációra vonatkozó megkötéseket, korlátozásokat írnak le.

Komponensdiagram:

komponens a rendszer fizikailag létező és kicserélhető része, ha a környezethez az új komponens csatlakozási felületét hozzáillesztjük.

Komponens

1. van azonosítója
2. objektumok egy osztálya ami a fizikai objektumokat definiálja
3. fizikai objektumok a szoftvermodulok fejlesztési v. végrehajtási formáiban léteznek.

   fejlesztési forma szöveges (fordítási kódforma)

   végrehajtási forma a modul végrehajtásra kész formája.
4. sztereo típus: létezési formát fejez ki.

komponensek közti relációk

1. fejlesztés során fennálló reláció
2. meghívási reláció

Környezeti diagram

rendszer topológiáját írja le.

Egységek megszorítása→pontosítjuk a leírást.

Használati esetek diagramja

felhasználók szempontjából kívánja szemléltetni a rendszer működését (szolgáltatás megvalósítás nem ismert).

- ▷ használati esetek
- ▷ felhasználók (lehetnek: programrendszerek, alrendszerek, osztályok, személyek)
- ▷ felhasználási relációk (használati eseteket összekapcsolják a felhasználókkal).
  - asszociáció
  - általánosítás
  - kiterjesztés
  - tartalmazás

20. A programtermék minőségi mutatói

- ▷ Programhelyesség
- ▷ megbízhatóság
- ▷ robosztusság
- ▷ Funkcionális minősítés
- ▷ Hatékonyság
- ▷ pszichológiai bonyolultság
- ▷ egyéb jellemzők

Programhelyesség Specifikáció-program kapcsolat minősítése

Megbízhatóság, robosztusság követelmények – program viszonyának minősége
Helyesség != megbízhatóság.
Időben változó, jellemezhetjük megbízhatósági fv-el.

Megbízhatósági fv. adott ideig nem nyújt hibás szolgáltatást.

Meghibásodási fv. hibás szolgáltatás nyújtásának valószínűsége.

Program bonyolultsága (architekturális bonyolultság)

- ▷ Ha nehéz a működését megérteni.
- ▷ Könnyű a megértés, ha
  - egyszerű, egymásra épülő absztrakciós szinten jól definiáltak
  - egységek közötti kommunikációs felület és az egység szolgáltatása elkülönülnek
  - architektúra elegánsan van megadva.
- ▷ Becslése: utasítások száma és/vagy döntések száma.