Rendszertervezés 3.

A rendszerfejlesztés tényezői

Dr. Szepesné Stiftinger , Mária

Rendszertervezés 3. : A rendszerfejlesztés tényezői

Dr. Szepesné Stiftinger , Mária
Lektor : Rajki , Péter

Ez a modul a TÁMOP - 4.1.2-08/1/A-2009-0027 „Tananyagfejlesztéssel a GEO-ért” projekt keretében készült. A projektet az Európai Unió és a Magyar Állam 44 706 488 Ft összegben támogatta.

v 1.0

Publication date 2010

Kivonat

A szoftverfejlesztés új technológiájának kialakulására részben hatékony munkaszervezési (menedzsment) módszerek, másrészt a nagy rendszerek uralására alkalmas módszertanok és programozási nyelvek fejlesztése hatott. Ez az átalakulás új elvek, módszerek és eszközök kifejlesztését, valamint szabványok bevezetését jelentette.

Tartalom

3. A rendszerfejlesztés tényezői 1
1. 3.1 Bevezetés 1
2. 3.2 Rendszerfejlesztés 1
3. 3.3 Az információrendszerek fejlesztésének időrendi modelljei, életciklus modellek 1
3.1. 3.3.1 VÍZESÉS MODELL 2
3.2. 3.3.2 SPIRÁL MODELL 3
3.3. 3.3.3 REKURZÍV/PÁRHUZAMOS MODELL 5
4. 3.4 Az információrendszerek fejlesztésének filozófiája (megközelítési módjai) 7
4.1. 3.4.1 Strukturált szemléletmód 7
4.2. 3.4.2 Objektum-orientált szemléletmód. 8
5. 3.5 Rendszerfejlesztési módszertanok (Eljárásrend) 10
5.1. 3.5.1 Strukturált módszertanok 11
5.2. 3.5.2 Objektum-orientált módszertan 14
5.3. 3.5.3 Az objektum-orientált rendszer tulajdonságai 15
5.3.1. Absztrakció 15
5.3.2. Hierarchia 16
5.3.3. Üzenetekkel való kommunikáció 18
5.4. 3.5.4 Rational Unified Process 19
6. 3.6 Összefoglalás 21

3. fejezet - A rendszerfejlesztés tényezői

1. 3.1 Bevezetés

A szoftverkrízis kialakulása után a nagy rendszerek uralására alkalmas módszertanok és programozási nyelvek fejlesztési feladata fontos kérdéssé vált, valamint a technikai aspektusokon túlmenően hatékony munkaszervezési (menedzsment) módszerek alakultak ki. Ekkor kezdődött el a szoftverfejlesztés technológiájának kialakulása. Az átalakítás **új elvek, módszerek és eszközök kifejlesztését, valamint szabványok bevezetését jelentette.**

Ebben a fejezetben az informatikai rendszerek fejlesztési tényezőiről lesz szó.

Megismerik a választ az alábbi kérdésekre:

- Az információrendszerek fejlesztésének időrendi modelljei?
- Mi az életciklus és milyen szakaszokra bontható?

Milyen életciklus-modelleket ismer? Miben különböznek ezek?

- Miért van szükség módszerekre, módszertanokra a rendszerek fejlesztésében?
- Leírjuk a strukturált szemléletmód lényegét és röviden ismertetjük az SSADM módszertant.
- Ismertetjük az objektumorientált szemléletmód lényegét és röviden a RUP módszertant.

2. 3.2 Rendszerfejlesztés

A szoftverkrízis megszüntetésének egyik eszköze a fejlesztés idejének szakaszokra bontását, életciklus modellek kialakítását várta el. A másik fontos eszköz a feladat, a rendszer tartalmának részekre bontása, ennek eredményeként létrejött a strukturált és objektumorientált szemléletmód, majd ezt követően a strukturált és objektumorientált módszertan. Mindkét területen a modellezés központi szerepet kap.

A szervezés a projekt szemlélet alapján történik.

A **projekt** egy olyan egyszeri feladat, amelyet adott idő és erőforrások felhasználásával az erre a feladatra szervezett munkacsoport hajt végre. Ez azt is jelenti, hogy minden egyes projekt egyszeri és megismételhetetlen, sohasem rutinmunka. A végrehajtásra szervezett csapat tagjai nem állandó jelleggel dolgoznak együtt, gyakran előfordul, hogy csak a munka megkezdésekor ismerik meg egymást. Rendszerint sok, különböző szakmabeli résztvevő munkáját kell összehangolni. A projektben egy meghatározott cél végrehajtása érdekében egymáshoz kapcsolódó tevékenységeket végeznek a résztvevők. (pl.: Házépítés) a folyamatok szereplői:

Szakterületi szakértők

IT szakemberek

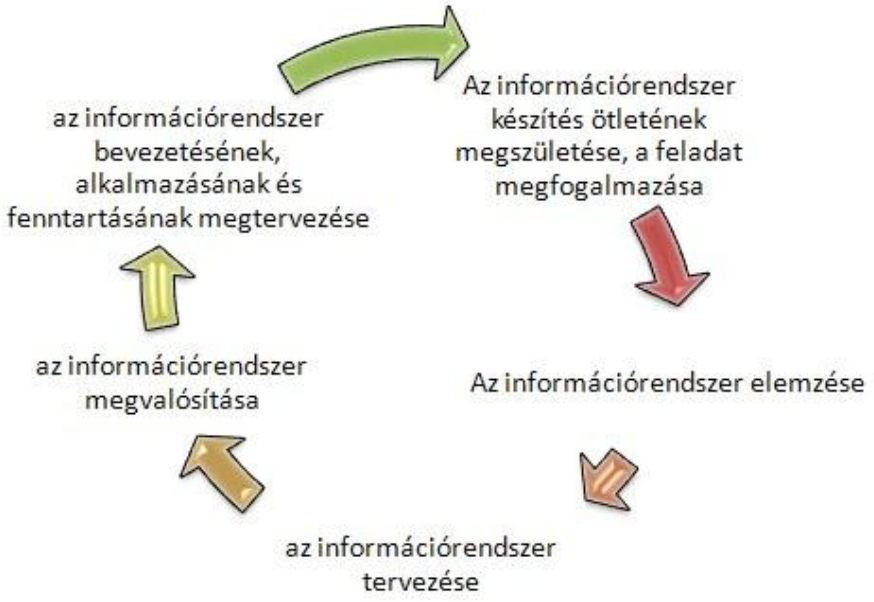
Projektirányítási, és minőségbiztosítási szakemberek.

A projekt során minden szereplőnek egyértelműen meghatározott feladata van, ezek ellenőrzését végzik a projektirányítási, és minőségbiztosítási szakemberek.

3. 3.3 Az információrendszerek fejlesztésének időrendi modelljei, életciklus modellek

A rendszerek elkészítése szigorúan meghatározott folyamatszakaszok során, gondos ellenőrzések mellett, részletes dokumentációval nyomon követve zajlik. Mindez természetesen már egészen a kezdetektől a

felhasználó, megrendelő bevonásával. Az információrendszer elkészítése időben több szakaszra bontható, ez jeleni a feladatok **időbeli ütemezését** . Ez több részfeladatot takar, a kész információrendszer elkészítési idejét, az egyes munkaszakaszokhoz szükséges idő tervezését, valamint az információrendszer erőforrásaira vonatkozó időbecslést. Feladat azt is meghatározni, hogy a kész információrendszer meddig nem igényel frissítést. Ezek a fázisok logikusan kapcsolódnak és kölcsönösen összefüggnek egymással.



3-1. ábra

Életciklus : Az igény felmerülésétől a termék használatból való kivonásáig (feledésbe merülésig) terjedő időtartam

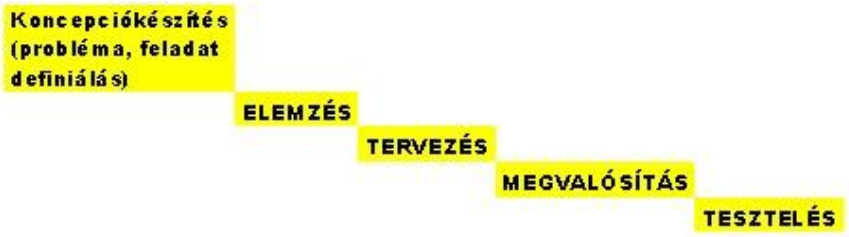
Az életciklus modellek:

Az életciklus modellek feladata, hogy segítsenek rendszerezetten átgondolni a folyamat egészét. A folyamatok egy rendszeren belül történnek, általában egy szervezeten belül és így lényeges szempont e folyamatok egységes (integrált) kezelése. Az egységes megközelítés akkor is igen fontos (nélkülözhetetlen), ha az adott rendszer egyes elemei kerülnek megtervezésre és kivitelezésre.

3.1. 3.3.1 VÍZESÉS MODELL

Egy lineáris lépéssorozat, amely az információrendszer fejlesztését elkülönülő, egymás után végrehajtásra kerülő fázisokra bontja. A fejlesztés során minden fázist csak egyszer hajtanak végre, visszalépést ez a modell a korábbi fejlesztési fázisok módosítására nem tervez. Ezen modell kapcsán alakult ki az életciklus fogalma.

A vízesés modellek szemlélete szerint a fejlesztés az egymásra épülő lépések előre pontosan meghatározott sorrendjéből áll.



3-2. ábra

A fázisok határait az ajánlások és a szabványok bizonyos dokumentációk (reprezentációk) meglétéhez, továbbá ezek áttekintését szolgáló összefoglaló és értékelő megbeszélések (**mérföldkövek**) megtartásához kötik.

A feladatok során keletkezendő szükséges kimenetek:

- követelményspecifikáció
- tervek
- ellenőrzött kód (megfelel a követelményeknek)
- integrációs eredmények

A vízesés modell használata során alkalmazott elvek:

- A célok legkönnyebben jól definiált és dokumentált mérföldkövek alkalmazásával érhetők el. Ezek a mérföldkövek a fejlesztést jól definiált szekvenciális szakaszokra bontják.
- A dokumentumok szerepe, érthetősége kulcsfontosságú.
- *A követelmények és a kívánt funkciók minden részlete ismert a fejlesztés megkezdése előtt, és ezek a fejlesztés során nem változnak.*
- Tesztelés és értékelés hatékonyan csak a fejlesztés végén kivitelezhető.

A gyakorlat mutatta, hogy a modellnek több problémás pontja van. Ilyenek a túlzott papíralapúság, az eredmények megjelenése túl sokáig tart, erősen függ a korrekt, nem változó követelményektől, a hibák csak a fejlesztés végén derülnek ki, bonyolult követni, hogyan alakulnak át a követelmények kóddá, szoftver-újrafelhasználást és a prototípuskészítést nem támogatja. E problémák miatt a modell csak korlátozottan használható.

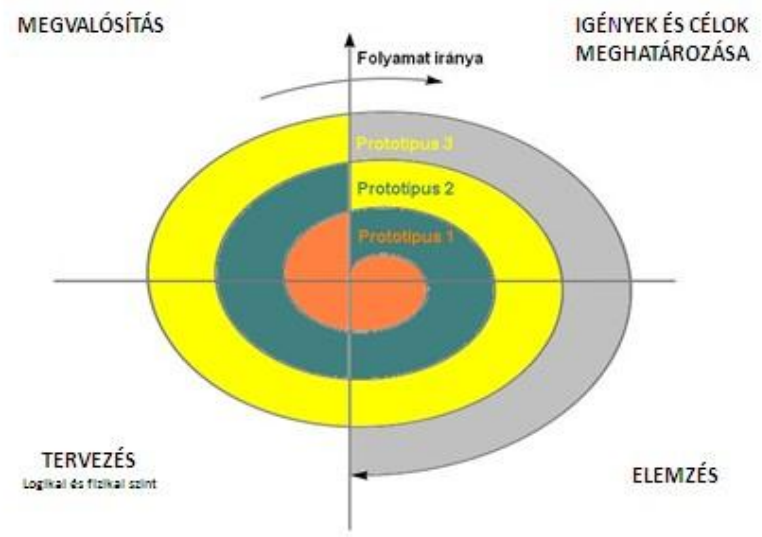
Ilyen modell valósul meg többek között az SSADM, a Yourdon, a Ward-Mellor metódusok használata során. Ezek közül talán legismertebb az SSADM.

3.2. 3.3.2 SPIRÁL MODELL

A spirál modellek szemlélete szerint a fejlesztés iteratív, újra és újra visszatér ugyanazokhoz a lépésekhez, és folyamatosan csiszolja a terveket, prototípusokat egészen addig, míg el nem ér egy, az igényeket maximálisan kielégítő termékhez.

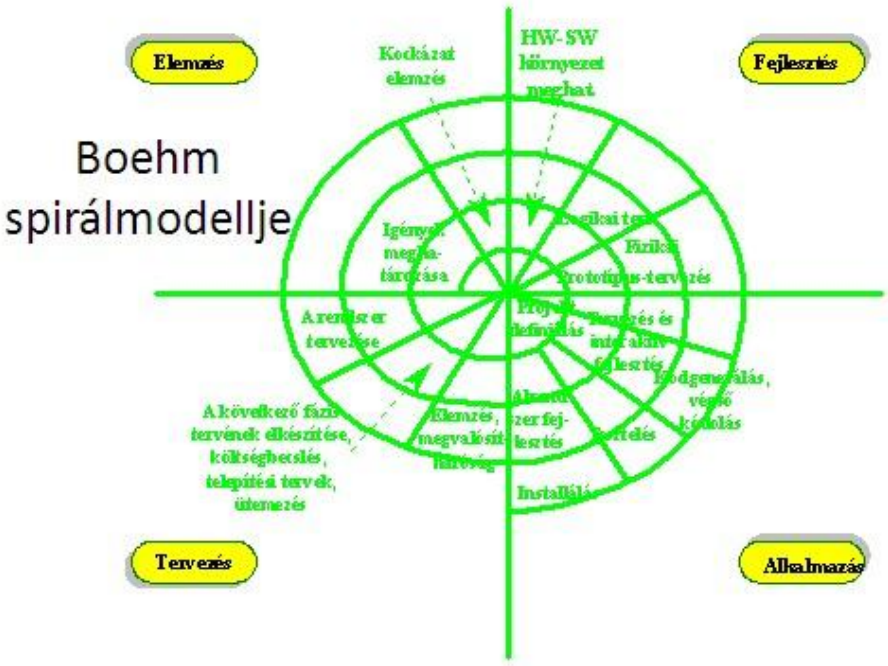
Prototípusok : A prototípusok fő jellemzője, hogy a tervbe vett rendszer főbb tulajdonságait viszonylag gyorsan bemutatják (még az előtervezési szakaszban), ami lehetővé teszi megoldási módok vizsgálatát, vagy a feladat megoldhatóságának bemutatását, és természetesen lehetővé teszi a továbbfejlesztést a teljes rendszer kialakulásáig.

A spirál modell ciklikusan ismétli a fejlesztési fázisokat. A spirál minden ága ugyanazon fejlesztési fázisokat tartalmazza, de a korábbi spirálágban elkészült változatot fokozatosan továbbfejleszti egységes szempontok alapján.



3-4. ábra

Részletesebben mutatja a fejlesztést a Boehm féle spirálmodell:



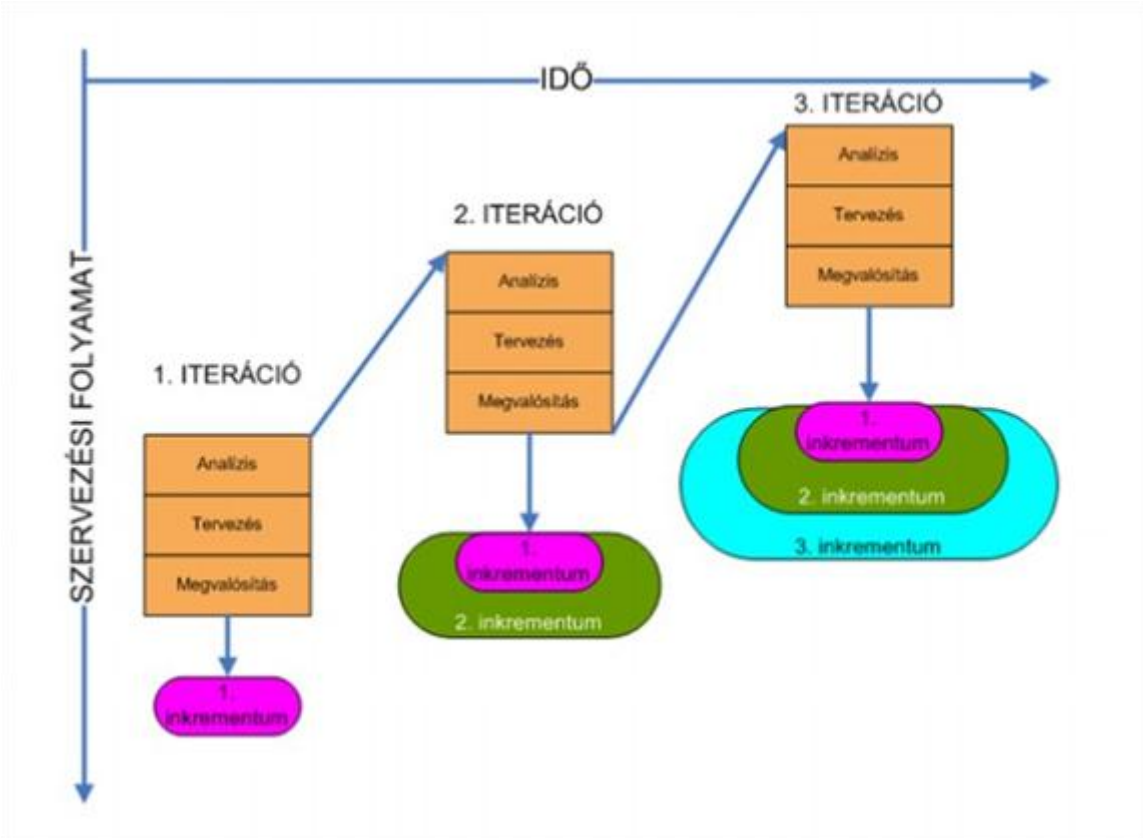
3-5. ábra

A két szemlélet között valójában nem olyan nagy a különbség, mint amekkorának az elsőre látszana, mert a vízesés modellek általában eleve magukban rejtenek néhány — pontosan rögzített számú — iterációt. A különbség lényege az, hogy *a spirál modell arra helyezi a legnagyobb súlyt, hogy folyamatosan ellenőrizze, korrigálja az előrehaladást, hogy a folyamat végén létrejövő termék biztosan megfeleljen az igényeknek.* Ezzel szemben a vízesés modell a folyamatot akarja pontosan megtervezni, és kontroll alatt tartani, elsősorban azért, hogy tudja, mennyi időre és pénzre lesz szükség a rendszerszervezés végig viteléhez.

Alapelvek a spirál modell alkalmazása esetén:

- Egy munkafolyamat a célok megértésével kezdődik, amelyek kockázatot is tartalmaznak.

- Az alternatív megoldások kiértékelése alapján azokat az eszközöket használjuk, melyek leginkább csökkentik a kockázatot.
- Minden érdekelt személyt vonjunk bele az áttekintés készítésébe, ugyanis ebben határozzuk meg a következő ciklus terveit és tevékenységeit.
- A fejlesztés minden szakaszban inkrementálisan folytatódhat.



3-6. ábra

A modell használata alatt a részkockázatok meghatározásával és kezelésével jó eredmények érhetőek el. A modell továbbfejlesztett változata (Transzformációs Modell) kiválóan alkalmas 4GL alkalmazások fejlesztésére.

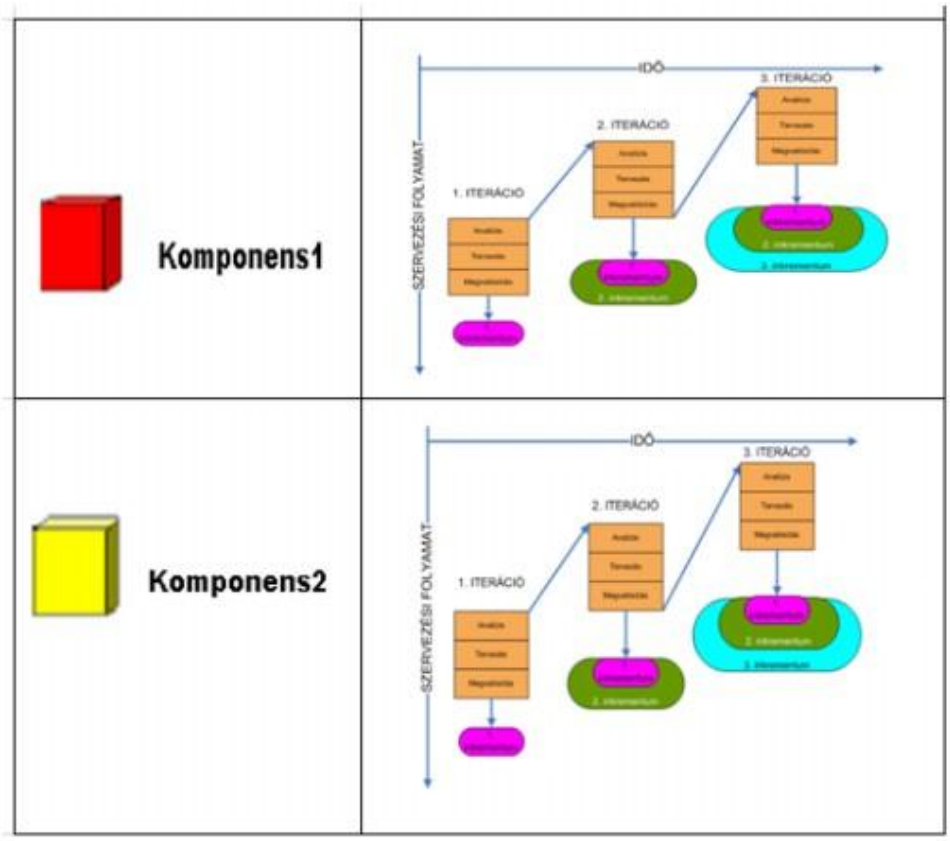
3.3. 3.3.3 REKURZÍV/PÁRHUZAMOS MODELL

A modell azon a felismerésen alapszik, hogy az elemzők és a tervezők minden fejlesztési fázisban egyszerre csak néhány követelményen dolgoznak, míg a többivel való teendőket későbbre halasztják. A modell első megközelítésében a problémát szisztematikusan egymástól független részekre bontjuk, majd minden komponensre a további dekompozíciót csak később végezzük el (rekurzív rész). A folyamat bármelyik komponens végrehajtásával szimultán folytatódhat. (párhuzamos rész). Minden folyamat, amelyik feldolgozásra kerül - egész vagy részekből álló – minden részére elemzés, tervezés, implementáció és tesztlépéseket kell végrehajtani.

A rekurzív/párhuzamos modell sikeréhez nélkülözhetetlen a komponensek lehetőleg egymástól független felbontása, amely miatt gyakran ajánlják ezt a modellt objektumorientált fejlesztéshez.

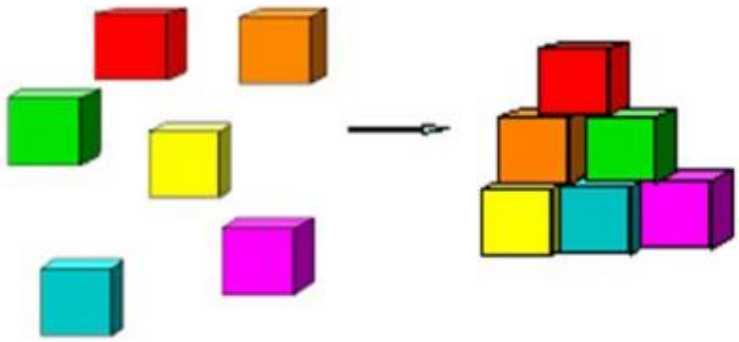
A főbb elvek, amelyeket a rekurzív/párhuzamos modell magába foglal:

- A fejlesztés központjában egymástól független komponenseken való párhuzamos munka áll.



3-7. ábra

- Az információrendszerek egymástól függetlenül előállított komponensekből összeállíthatóak.



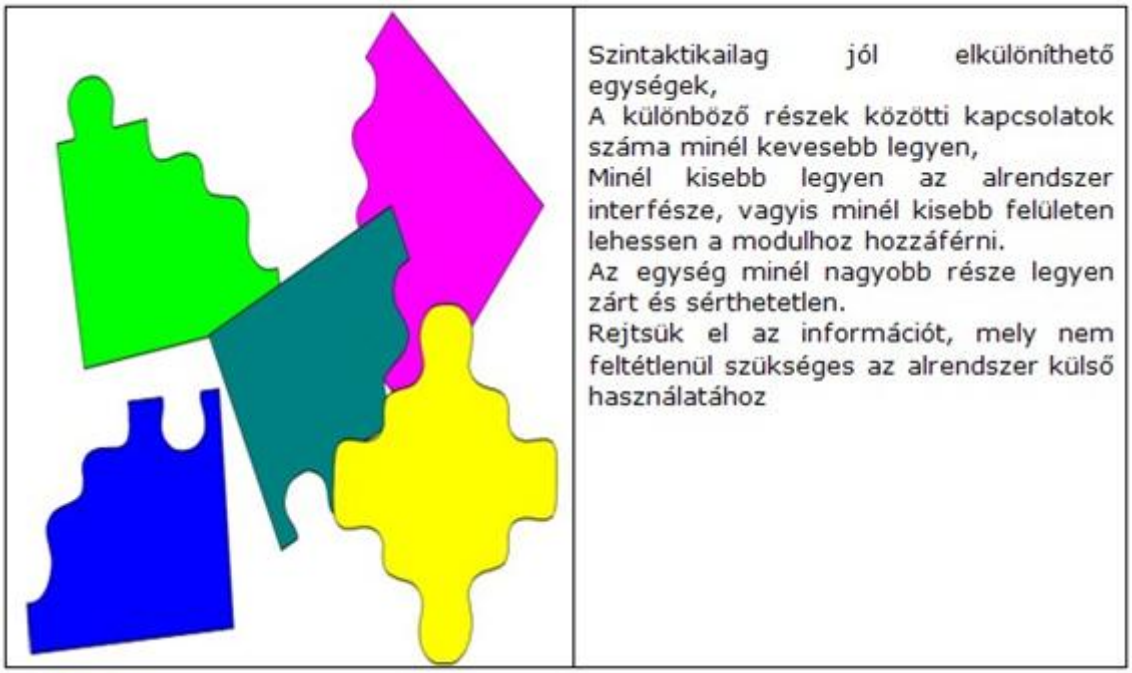
3-8. ábra

Általánosan az OOA/D különböző fejlesztési szinteket különböztet meg:

- Tárgyi szintet - a probléma meghatározása, kategóriák, funkcionalitások szétválasztása,
- Objektum szintet - objektum és osztályazonosítás, meta-osztályok, egyedek, terminátorok,
- Strukturális szintet - hierarchiák és függőségek objektumok között,
- Szervíz szintet - metódusok, üzenetek, adat folyamok.

4. 3.4 Az információrendszerek fejlesztésének filozófiája (megközelítési módjai)

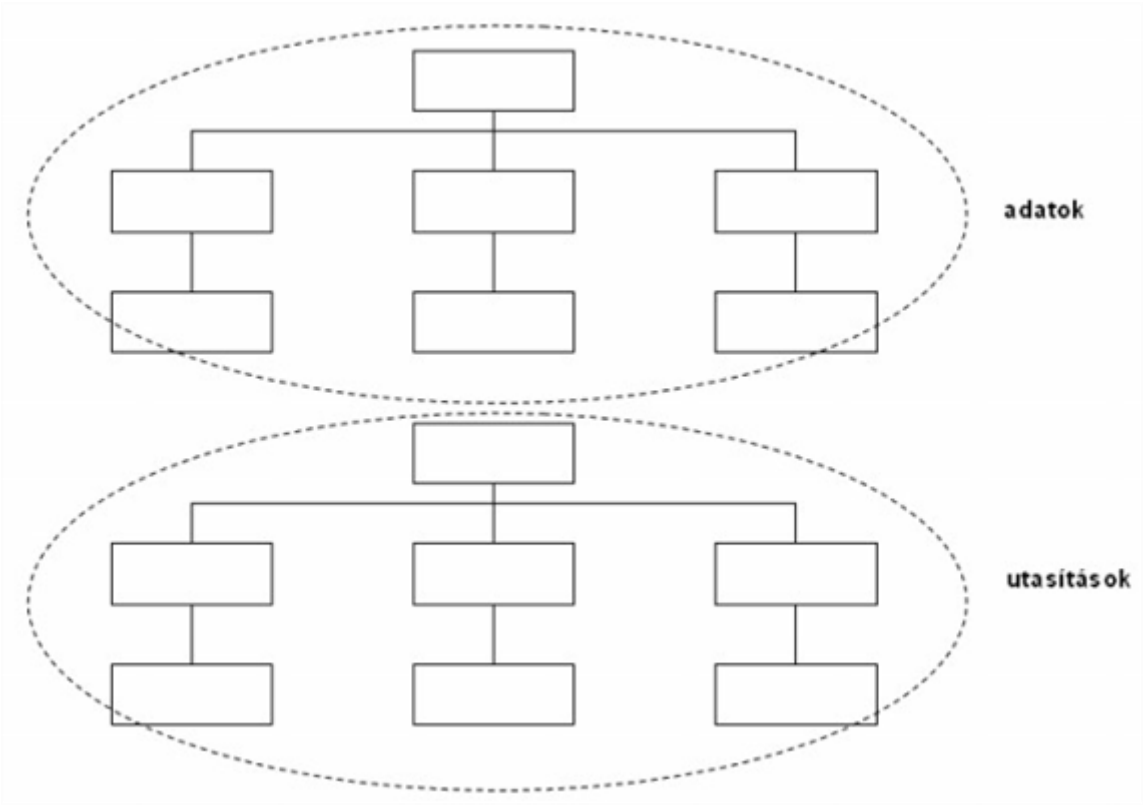
A rendszer szerkezeti elemzése módszertani-technikai lehetőséget ad ahhoz, hogy kisebb egységekben gondolkodjunk. A rendszer elemeit és azok viszonyait többé-kevésbé egymástól függetleníthető csoportokba soroljuk.



3-9. ábra

4.1. 3.4.1 Strukturált szemléletmód

A feladatok felbontása két szinten történik, részben rendszer alrendszerekre bontása, majd az alrendszerek adat és az adatokon végzett műveletek különválasztását jelenti. A strukturált fejlesztés során a logikai tervezésnél élesen különválik az adatok és folyamatok tervezési szakasza. A folyamatok, műveletek: eljárások és függvények.



3-10. ábra

Az adatok és a műveletek között a kapcsolatot paraméter átadással valósítják meg.

Jellemzői:

• Sem a folyamatoknak sem az outputoknak, sem az adatoknak nincs elsőbbsége.

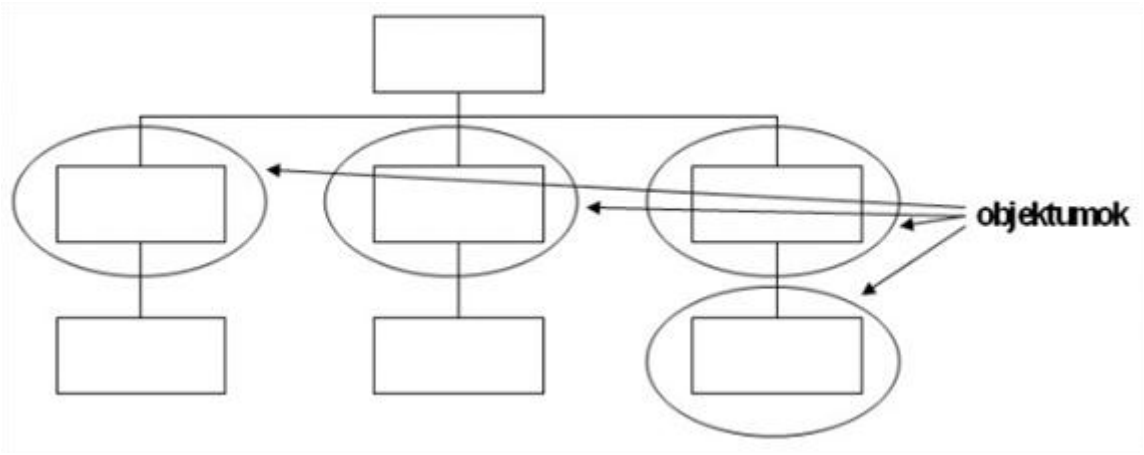
• Programokat sokkal könnyebben el lehet készíteni, ha modulokra bontjuk őket, és a modulokat összekapcsolva egy szerkezetet, struktúrát alakítunk ki.

• A strukturálásra szabályokat lehet kidolgozni, ezen alapul a strukturált programozás elmélete és gyakorlata is.

A felbontást követően a rendszerfejlesztés elemenként történik, majd az elemekből a struktúra alapján építjük fel a rendszert.

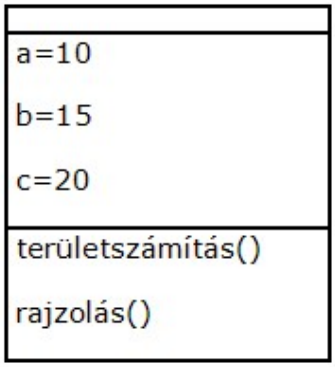
4.2. 3.4.2 Objektum-orientált szemléletmód.

Az emberi gondolkodáshoz közelebb álló szemléletmód. Emberek milliói képesek autót vezetni anélkül, hogy pontos képük lenne arról, mi történik a motorház fedele alatt. Az autó számukra jól definiált kezelői felülettel rendelkező objektumok csoportja, amelyek az emberrel és egymással együttműködve összességében "autó"-ként viselkednek. Az autó használata szempontjából nem különösebben érdekes, hogy a motort benzines, dízeles vagy elektromos energia működteti.



3-11. ábra

Az objektum fogalmának lényege az a felismerés hogy a környező világ dolgai jobban modellezhetők számítástechnikai eszközökkel, ha jellemző adataikat és működési módszereiket nem egymástól elválasztva, hanem egységes egészként kezeljük. Az emberek a világ dolgait objektumokként kezelik (ház, ember, iskola, egyetem, háromszög).



3-12. ábra

Ezen egységek (elemek) a világ dolgainak megismerése során az emberi gondolkodás révén alakulnak ki.

A gondolkodás eszközei:

Absztrakció: egyszerűsítés, csak a lényeg figyelembe vétele, pl.: modellezés, mint például térképek készítése.

Megkülönböztetés: meg tudjuk különböztetni a dolgokat a számunkra lényeges tulajdonságok alapján, pl.: piros alma, zöld alma.

Osztályozás: a fontos tulajdonságok alapján kategóriákba soroljuk a dolgokat, pl.: növény, gyümölcs, alma, piros alma.

Általánosítás, specializálás: hasonlóságok keresése, új kategóriák létrehozása, pl.: akik tanulnak ők a diákok, akik a felsőoktatásban tanulnak ők a hallgatók, akinek jók a jegyei az jó tanuló.

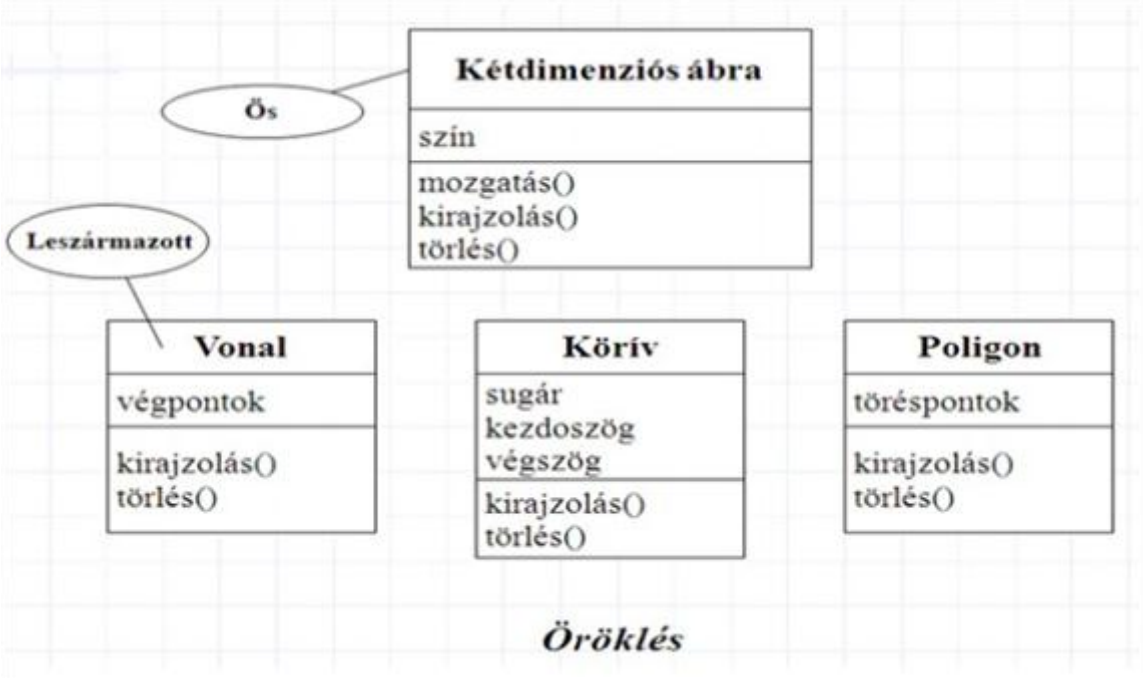
Kapcsolatok definiálása: a dolgok közötti kapcsolatok keresése, meghatározása.

Az OO szemléletmód jellemzői:

Az **egységbezárás** (*encapsulation*) azt jelenti, hogy az adatstruktúrákat és az adott struktúrájú adatokat kezelő függvényeket (metódusokat) kombináljuk; azokat egy egységként kezeljük, és elzárjuk őket a külvilág elől. Az így kapott egységeket **objektumoknak** nevezzük, pl.:

Az egységbezárás vagy becsomagolás (encapsulation), vagyis az alapelemek belső állapotának elrejtése és interaktív tulajdonságainak megadása egy működési interfésszel (azoknak az eseményeknek a definiálásával, amelyekben ez az elem részt tud venni).

Az **öröklés** (*inheritance*) azt takarja, hogy a meglévő objektumokból levezetett újabb objektumok öröklik a definiálásukhoz használt alap objektumok egyes adatstruktúráit és függvényeit; ugyanakkor újabb tulajdonságokat is definiálhatnak, vagy régieket újraértelmezhetnek. Olyan, osztályok közötti viszony, amely lehetővé teszi, hogy egy osztály sajátjaként kezelje a nála általánosabb osztályban definiált attribútumokat és műveleteket. Az általános osztályt szuper vagy szülő-osztálynak, a speciális tulajdonságokkal rendelkezőt pedig szub, vagy gyerek osztálynak nevezzük.



3-13. ábra

A **többalakúság** (*polymorphism)* alatt azt értjük, hogy egy adott tevékenység (metódus) azonosítója közös lehet egy objektum hierarchián belül, ugyanakkor a hierarchia minden egyes objektumában a tevékenységeket végrehajtó metódus implementációja az adott objektumra nézve specifikus lehet. Többalakúság: az objektumok azon képessége, amely lehetővé teszi adatainak, illetve műveleteinek egymástól eltérő formában való megjelenését, vagyis egy objektum több különböző formát is képes ölteni attól függően, hogy milyen hatás éri.

újrafelhasználhatóság : a fejlesztés során a különböző szinteken kidolgozott modell-elemeknek többszörös, ismételt felhasználási lehetősége, akár más fejlesztésekben, vagy más alkalmazásokban.

5. 3.5 Rendszerfejlesztési módszertanok (Eljárásrend)

Módszer a megismerés tudatosan szabályozott eszköze. Valamely eredményhez elvezető tervszerű eljárás. Alapját a valóság objektív törvényei alkotják. Módszertan az adott területhez, témakörhöz tartozó módszerek összessége.

Az információrendszerek fejlesztése során a módszertan az informatika sajátosságainak megfelelően kibővül a tudományág speciális eszközeivel.

Módszertan:„elvek, módszerek és fejlesztést támogató eszközök, technikák egysége”

Módszertan használata során, minden esetben rögzíteni kell a feladatot, módszert, terméket.

3-14. ábra

Módszertan alapján készített IR-ek esetén:

Nincs módszertani vita, üresjárat,

Technologizált munka,

Pontos elvárások,

Követhető, ellenőrizhető,

Definiált termékek.

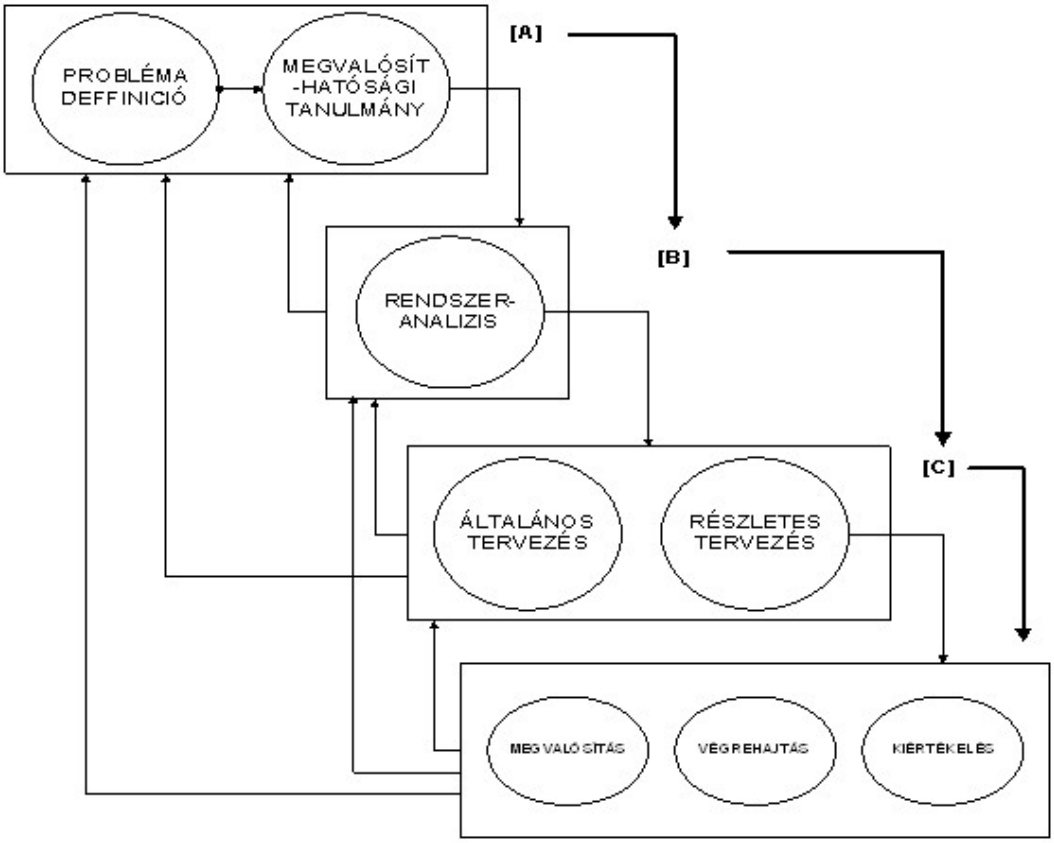
Csereszabatosság.

5.1. 3.5.1 Strukturált módszertanok

A strukturált módszertanok felhasználják a korábbi eredményeket, párhuzamosan megjelenő eredményeket, a strukturált programozás megjelenését, a projektvezetési módszertan kialakulását és a dokumentáció jelentőségének felismerését.

Strukturált módszertanok legfőbb jellemzői a következők:

Életciklus modell

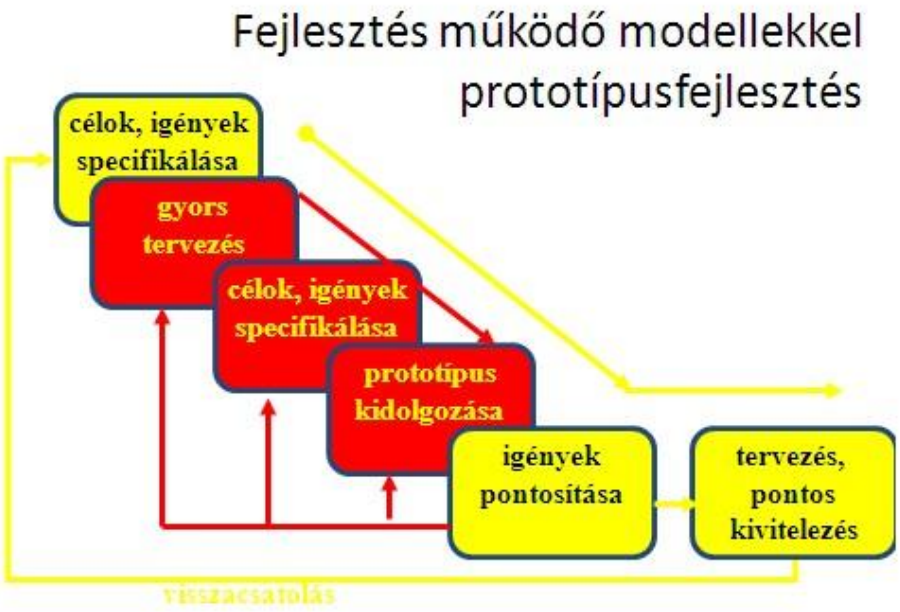


3-15. ábra

A rendszerek életciklusa rendkívül hosszú, így létrehozásuk, továbbfejlesztésük csak szigorúan egymásra épülő fázisokban lehetséges. A strukturált rendszerépítkezés és a prototípus-rendszer (pilot projekt) elterjedése nagymértékben segíti a felhasználói igényeknek megfelelő informatikai környezet kialakítását.

Prototípus-rendszer (pilot projekt MINTA FELDOLGOZÁS)

A prototípusok fő jellemzője, hogy a tervbe vett rendszer főbb tulajdonságait viszonylag gyorsan bemutatják (még az előtervezési szakaszban), ami lehetővé teszi megoldási módok vizsgálatát, vagy a feladat megoldhatóságának bemutatását, és természetesen lehetővé teszi a továbbfejlesztést a teljes rendszer kialakulásáig. Azért készítik, hogy felismerjék az előre nem látható problémákat, amelyet a rendszertesztek nem mutatnak ki, kialakítsák a végleges tervet és az adat-átalakítási eljárásokat. A pilot projekttel megismerhető a különbség a rendszer működése és leírása között.



3-16. ábra

A strukturált módszertanok *szabványosítják* a fejlesztés menetét, a fejlesztés szakaszait, az azon belül folytatandó tevékenységeket. Ezzel a módszertanok a projektvezetésnek nyújtanak kiindulási alapot.

Termékszemlélet

A strukturált módszertanok előírják, hogy a fejlesztés egyes lépéseinek milyen dokumentumok a termékei. Ezek az előírások a projektvezetésnek és a minőségbiztosításnak nyújtanak kapaszkodási pontokat.

Technikák

A strukturált módszertanok pontos és részletes útmutatást adnak arra, hogy az egyes lépésekben az egyes termékeket hogyan, milyen technikával kell előállítani.

Fizikai és logikai szint szétválasztása

Fokozatosság és iteráció

A fentieket megvalósító néhány strukturált módszertan: SUMMIT-D (*Coopers & Lybrand*), Method/1, (*Andersen Consulting*), SDM (Pandasoft), SSADM (*CCTA, NCC*)

Az SSADM = Structured System Analysis and Design Metod (Strukturált Rendszer Elemzési és Tervezési Módszer)

A módszer a 80-as évek közepétől kormányzati szabvánnyá vált. 1993 óta az Informatikai Tárcaközi Bizottság ajánlataként a magyar kormányzati szervek is alkalmazzák. Az SSADM szerkezete, hierarchikus felépítése és termékközpontúsága lehetővé teszi a feladatok, termékek, határidők, ellenőrzési pontok hatékony kezelését.

Az SSADM három nézőpontja:

Funkciók : A funkciók a felhasználók nézeteit tükrözik az eseményekre reagáló rendszer-feldolgozási folyamatokról.

Események : Az események lehetnek a működési terület valós eseményei, mint például lakáskeresés feltételeit tartalmazó űrlap elküldése, vagy olyan rendszer által indított események, mint például egy automatikus email generálása, a feltételeknek megfelelő lakás megjelenítése.

Adatok : A rendszer adatokat kezel és tart karban annak érdekében, hogy nyújtani tudja a rendszer funkcionalitását.

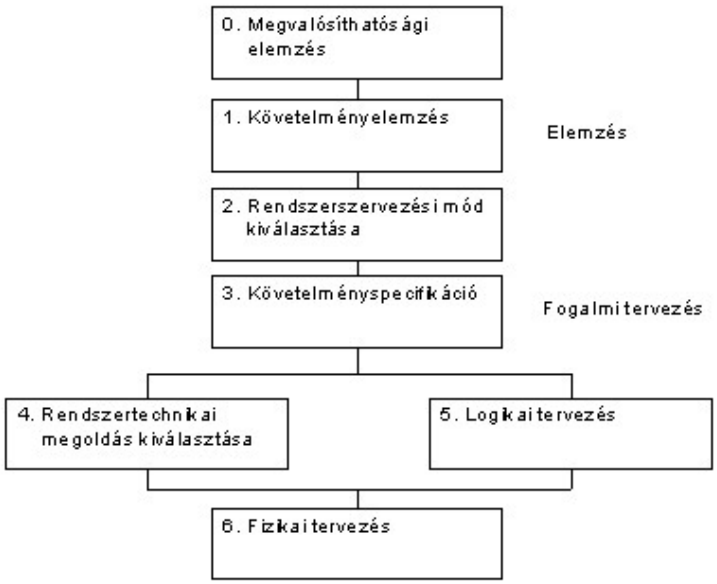
A követelményeket mind a három perspektívából meg kell határozni, bármelyik elhagyása azt eredményezheti, hogy a rendszer-követelmények teljességét nem sikerül átfogó módon nyújtani.

A három nézetnek megfelelően az SSADM alapja:

- az adatok logikai modellje *(logikai adatmodell)*
- folyamatok, adattárak és külső egyedek közötti adatáramlás modellje *(adatfolyam-modell)*

egyedeket módosító adatfolyamok eseményeinek hatását leíró modell *(egyed-esemény modellek)*

SSADM modulok és szakaszok:



3-17. ábra

A tervezési folyamat hármas felosztása

- *Fogalmi modellezés* elemeinek tekintjük a logikai adatmodellt, valamint az eseményeket, lekérdezéseket. Azért tekintjük ezeket fogalminak, mert függetlenek mind a hardver-szoftver környezettől, mind pedig attól, hogy hogyan kell a tervezett rendszernek megjelennie a felhasználó előtt.
- *Belső tervezés* a fogalmi modell elemeinek a konkrét adatkezelő és feldolgozó eszközökre való leképezését jelenti. Nem azonos a fizikai tervezéssel.
- *Külső tervezés* a fogalmi modell és a felhasználók között teremt kapcsolatot. Ide tarozik mindannak a megtervezése, ami a felhasználói interfész alkotóelemének tekinthető.

5.2. 3.5.2 Objektum-orientált módszertan

Az objektumorientált módszertan alkalmazásával a kifejlesztendő rendszert együttműködő objektumokkal modellezzük, a tervezés és az implementáció során pedig ezen objektumokat "szimuláló" programegységeket alakítunk ki.

Az objektum-orientált (O-O) rendszerek objektumnak nevezett blokkokból épülnek fel. Minden ilyen elem tartalmazza azon műveleteket és adatokat, amely szükséges egy adott cél eléréséhez. Egy objektum végrehajt egy feladatot, amikor erre utasítást kap. Mivel ilyen módon "zárt", az objektum akárhány programon belül újrahasznosítható, mint egy teljes egység. Az objektum-orientált módszertanok a valós világ modellezése során az objektumokat és ezek kölcsönhatását más objektumokkal veszik figyelembe. A gyakorta ismétlődő dolgoknak a szoftver objektumokban történő modellezésével az újrafelhasználhatóság irányába is jelentős lépést tehetünk. Az objektumorientáltság a szoftverkrízis leküzdésének egyik lehetséges eszköze.

Egy objektum-orientált rendszer jellemzői:

- A rendszer elemei: az objektumok - Szerkezet és viselkedés egységbezárása
- Az alkalmazások szilárdsága - részletek elrejtése
- Újrafelhasználható integritások - általánosított megoldások
- Újrafelhasználási módok (példányosítás, öröklődés, polimorfizmus)

5.3. 3.5.3 Az objektum-orientált rendszer tulajdonságai

5.3.1. Absztrakció

Az absztrakció az az eljárás, melynek során egy komplex valós világból vett helyzetet egy egyszerűsített modellel "helyettesítünk", úgy, hogy a kiválasztott helyzet, dolog tulajdonságai közül csak azokat vesszük figyelembe, amelyek a cél elérése érdekében feltétlenül szükségesek. Csak a **lényegre** koncentrálunk. Az **osztályozás** a természetes emberi gondolkodás szerves része. Az objektum-orientált technológiákban az emberi gondolkozáshoz hasonló folyamat kerül végrehajtásra, amelynek eredményeként osztályokat hoznak létre.

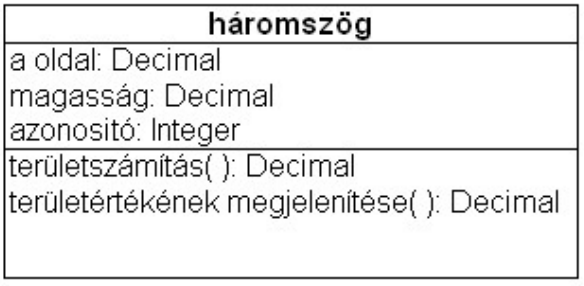
Osztályok: Az ugyanolyan adatokat tartalmazó, és az ugyanolyan viselkedés-leírással (metódusokkal) rendelkező objektumokat egy **osztályba** soroljuk. Az objektum-osztályok hordozzák a hozzá tartozó objektumok jellemzőit. Minden objektum valamilyen osztály példánya (instancia), rendelkezik osztályának sajátosságaival, örökli annak tulajdonságait az adatszerkezetre és a műveletekre vonatkoztatva egyaránt. Általában elkerüljük a nevek használatát egyedi objektumok meghatározása esetén, mivel általában az objektumoknak nincs természetes nevük. E helyett a leírásokat használunk azért, hogy végül is egyedi entitásokat fejezzenek ki. Az objektumok jellemzői fogják elvégezni ezen leírást.

Az **osztály** az azonos sajátosságokkal rendelkező, a rendszer célja szempontjából összetartozó objektumok összessége, amelyek attribútumokkal és műveletekkel jellemezhetők. Az osztály olyan fogalmi kategória, amely a valós világ konkrét dolgainak (személyek, cégek, termékek stb.), az objektumoknak valamilyen szempontból egységes, absztrahált halmazát reprezentálja.

Az osztály meghatározott , ha

- névvel rendelkezik, névvel egyértelműen hivatkozhatunk rá,
- véges számú attribútummal jellemezhető, ahol az attribútumokra egyértelműen tudunk hivatkozni,
- azonosítója van, amely alkalmas az osztály előfordulásainak (instanciáinak) meghatározására,
- ismert az előfordulások (instanciák) halmaza.

OSZTÁLY:



3-18. ábra

Attribútumok: objektumban tárolt adatok , amelyek **az objektum tulajdonságait és állapotát határozzák meg.** A valós élet dolgai gyakran stabil jellegzetességet mutatnak. A legtöbb természeti objektumnak vannak jellegzetességei, mint például az alakja, súlya, színe és anyag típusa. Az embereknek is vannak jellegzetességei, amelyek közé tartoznak, pl. a születési dátum, szülők, név és a szemek színe. Az attribútumok száma és típusa definiálja az objektum struktúráját, pl. Ember objektum attribútumai lehetnek: név, személyi szám, cím, havi jövedelem, autó rendszáma stb.

Az attribútumokat csoportosíthatjuk aszerint, hogy milyen típusú információkat tartalmaznak. Így lehetnek:

Elnevezés típusú - az az attribútum, amelyik az adott objektum valamely jellemzőjének megnevezését, címkéjét tartalmazza, pl. név, személyi szám. Ezek az attribútumok az objektum élete során általában nem változnak.

Leíró típusú - az objektum olyan belső jellemzőit rögzítik, amelyek az objektum élete során, az objektumot ért hatások következtében változnak, és az objektumnak valamilyen kiértékelhető tulajdonságát adják meg. Minőségi jellemzőnek is nevezhetjük. Pl. cím, havi jövedelem.

Referenciák - más objektumokra hivatkoznak. Az objektumok közötti kapcsolatokat, relációkat valósítják meg.

Tehát egy osztály meghatározása jellemzői vagy tulajdonságai segítségével történik. A tulajdonságok csoportja nem változik.

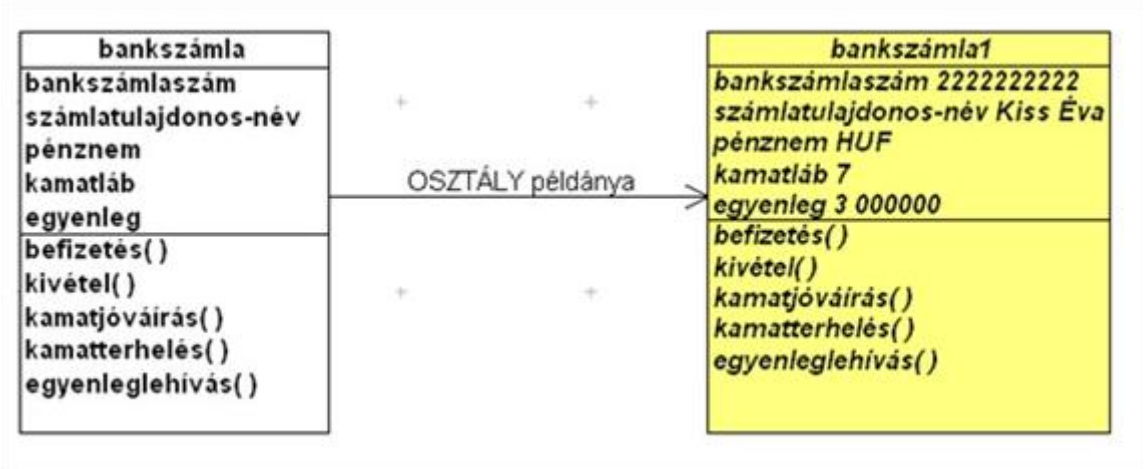
Az attribútumok szerepük szerint lehetnek:

- az osztály egy adott objektumának azonosítása **azonosító** attribútum: az osztály egy adott példányának megkeresésére szolgál
- az objektum leírása, jellemzése **leíró** attribútum
- **kapcsoló** attribútum: biztosítja a kapcsolatot az osztály egy adott objektumának egy másik osztály objektumával

A **művelet** az objektumok által végrehajtott tevékenység, olyan fogalmi kategória, amely az esetek többségében az objektumok állapotváltozását eredményezi. **A metódus** : egy adott művelet megvalósulása, olyan lépéssorozat, amely konkretizálja a műveletet. Két műveletcsoport:

- *módosító:* megváltoztatják az adott objektum állapotát,
- *lekérdező:* ezek paraméterértéket kérnek más objektumoktól.

Minden objektum valamilyen osztály példánya (instancia), rendelkezik osztályának sajátosságaival, örökli annak tulajdonságait az adatszerkezetre és a műveletekre vonatkoztatva egyaránt.



3-19. ábra

Objektumok, mint osztályok példányai : Mi akkor használunk példányokat, ha ki akarjuk hangsúlyozni, hogy egy objektum egy osztály példánya. Időnként szükségünk van egy példányról beszélni a rendszermodellünk keretein belül. Általában viszont objektum gyűjteményeket, úgynevezett osztályokat használunk.

5.3.2. Hierarchia

A rendszer elemei az osztályok, a közöttük lévő kapcsolatok alapján építhető fel a rendszer, azaz adható meg a rendszer struktúrája, szerkezete. A felépítés, hierarchia az osztály diagram segítségével modellezhető.

Az objektum-orientált tervezésben az objektumok osztályai, az osztályok közti kapcsolatok alapján hierarchiákba vannak elhelyezve, amelyek modellezik az osztályok közti viszonyt.

Az osztályok közti relációk:

Ismeretségi vagy használati kapcsolat (**Asszociáció**):

Két osztály közötti kapcsolatot jelent,

a két osztály léte független,

legalább az egyik ismeri/használja a másikat,

mindkét irányban lehet információ/adat továbbítás

A valódi összefüggés az osztályokból létrejövő objektumok között jön létre. Az asszociáció azt fejezi ki, hogy a két (esetleg több) osztály példányai kapcsolatban vannak (vagy lehetnek) egymással.

Amennyiben a reláció pontosan két objektumot (osztályt) kapcsol össze, akkor **bináris relációról** beszélünk. Az elemzés során találkozhatunk kettőnél több osztály között fennálló relációval, de ezek többsége alapos vizsgálat után szétszedhető bináris relációkká. Elvétve előfordul, hogy három osztály kapcsolata nem bontható fel információvesztés nélkül – ezt **ternáris relációnak** hívják.

Az asszociáció multiplicitása megadja, hogy az asszociáció az egyik osztály adott példányát a másik osztály (amelyikre *az asszociáció irányul) hány példányával kapcsolja, vagy kapcsolhatja össze.*

A kapcsolat megvalósulásánál résztvevő objektumok számát írjuk le.

Példák:

1 pontosan egy

0..* 0 vagy több

1..* 1 vagy több

0..1 0 vagy 1

Tartalmazási kapcsolat , mely az egész-rész viszonyt tükrözi (más osztályokat tartalmaz, mint egy adott osztály részeit). Az egészet reprezentáló osztály az aggregált, a rész pedig a komponens.

Fajtái:

aggregáció csak akkor,

- ha tényleg rész-egész kapcsolatról van szó
- ha a rész nem értelmes az egész nélkül

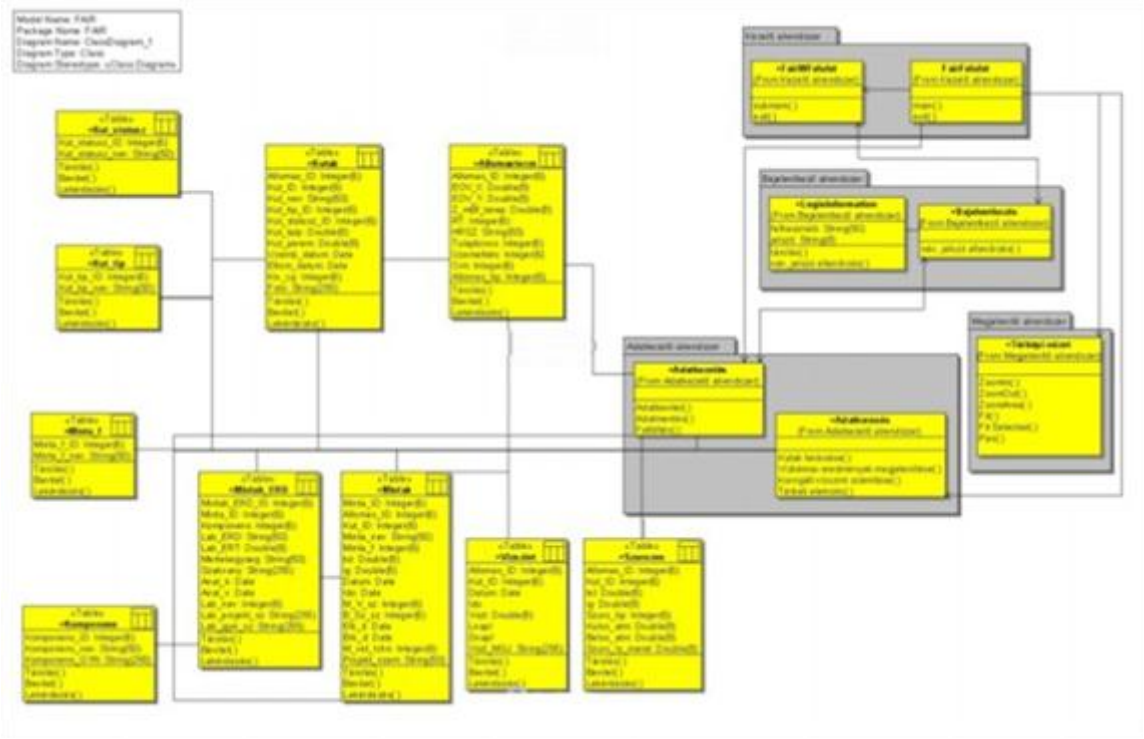
kompozíció : szoros kapcsolat a rész és az egész között:

- az egész megszűntével a rész is megszűnik
- az egész nem teljes a rész nélkül
- az egész oldal számossága csak 1 lehet

Osztálydiagram (class): a rendszer objektumelvű szerkezetének leírása. Az osztálydiagramok legalapvetőbb objektumorientált modellező eszközök, melyekkel a rendszert fölépítő objektumokat és a közöttük lévő statikus kapcsolatokat írhatjuk le.

PÉLDA:

Egy földtani IR fejlesztéséhez tartozó osztálydiagramot mutat az alábbi ábra, mely földtani IR egységes rendszerbe foglalja a földtani alapadatokat és a kutatások köré csoportosuló mérési adatokat, lehetőséget biztosít az adatok tárolására, feldolgozására, elemzésére és szükség szerinti szolgáltatására, illetve megjelenítésére.



3-20. ábra

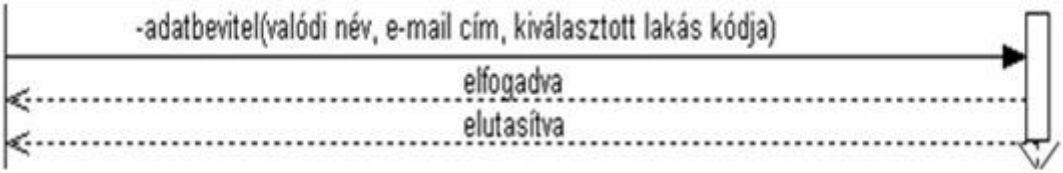
CSOMAG Diagram

A funkcionálisan összefüggő modellelemek egyetlen magasabb szintű egységbe foghatók, így rendszerünk statikus struktúrája kezelhetővé válik. A csomagok alapvetően osztályokból állnak, az osztályok között pedig függőségek léteznek, ha egymástól függő osztályok különböző csomagokba kerülnek, akkor ez a csomagok közötti függőségek, kialakulásához vezet.

A komponensek szoftvermodulok fizikai kódját testesítik meg, melyek a csomagdiagramokban megjelenő csomagoknak felelnek meg. A komponensek közötti kommunikáció a megfelelő csomagok közötti függőségek alapján történik.

5.3.3. Üzenetekkel való kommunikáció

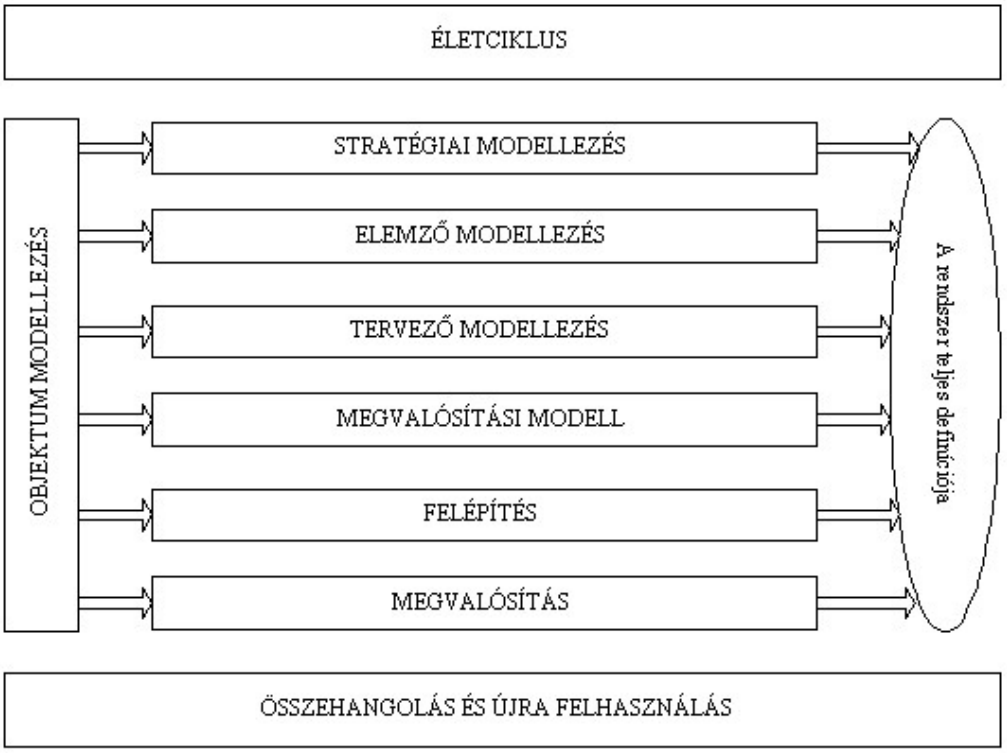
Az objektumok kölcsönhatásban vannak más objektumokkal, amelyek ugyanazon vagy más osztályba tartozhatnak. Ez az által valósul meg, hogy üzeneteket küldenek egymásnak. Így információt adnak át, vagy egy adott hatást kiváltását kérik. Például, amikor a felhasználó kiválaszt egy vezérlő gombot a dialógus során, egy üzenetet küld a dialógus objektumnak, melyben közli, hogy a vezérlő gomb le volt nyomva.



3-21. ábra

Az OO módszertan a szoftverek teljes életciklusát felöleli.

Fázisok: Koncepciókészítés (probléma, feladat definiálás), elemzés, a rendszer tervezése, objektum-modell tervezése, megvalósítás, tesztelés, rendszerkövetés.



3-22. ábra

Az OO módszertan jellemzői:

- modellszemléletre épít (valóságnak megfelelő modell létrehozása a modellezés alapkérdése),
- hatékonyan valósítja meg a rendszer alrendszereinek és ezen alrendszerek együttműködésének modellezését,
- a modell leírása UML szimbolikával történik,
- a minőségi követelmények teljesítése könnyebben megvalósítható,
- egyszerű, gyors módosíthatóság jellemzi,
- az újrafelhasználhatóság kézenfekvő.

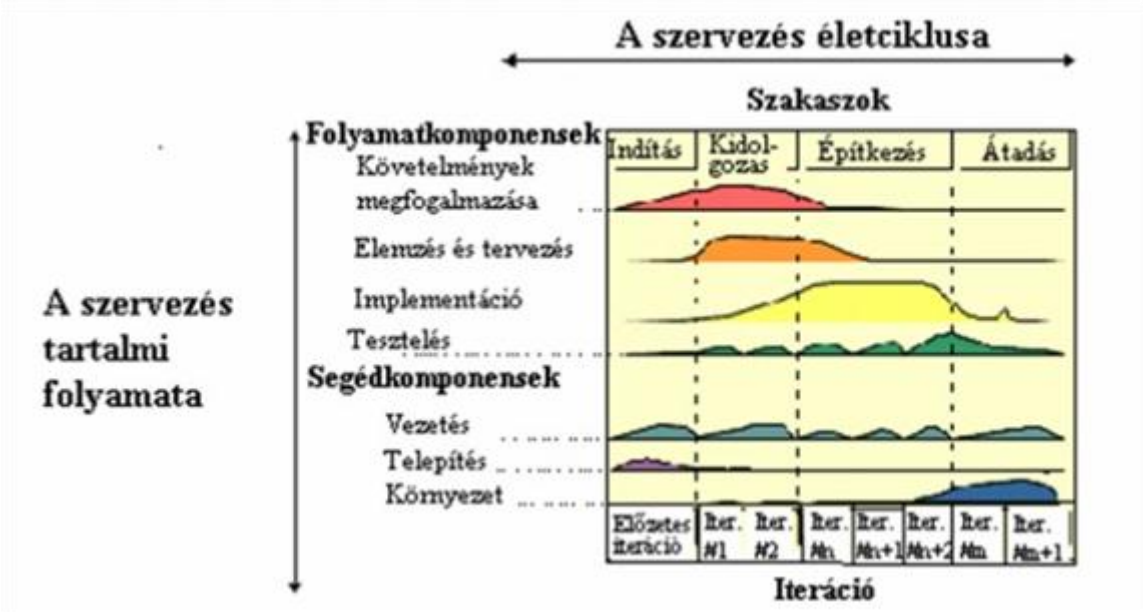
OO fejlesztési módszertanok:

- OMT (Object Modelling Technique,Rumbaugh, 1991, majd 1993)
- OOD (Object Oriented Design, Booch, 1991, majd 1993)
- OOA (Object Oriented Analysis, Coad &Yourdon, 1991)
- OOSD (Object-Oriented Structured Design, Wasserman, 1990)
- HOOD (Hierachical Object Oriented Design, 1989)
- Responsibility -Driven Design, Wirfs & Buck, 1990
- OOSE (Object Oriented Software Engineering, Jacobson, 1992
- RUP (Rational Unified Process) (UML + fejlesztési folyamat ajánlás, 1998-1999

5.4. 3.5.4 Rational Unified Process

A Rational Unified Process (RUP – „Rational” egységesített eljárása) a Rational (később IBM) szoftverfejlesztési módszertana. Definíciója szerint a módszertan " *elvek, módszerek és fejlesztést támogató eszközök, technikák egysége* ". *A RUP legfőbb jellemzői:*

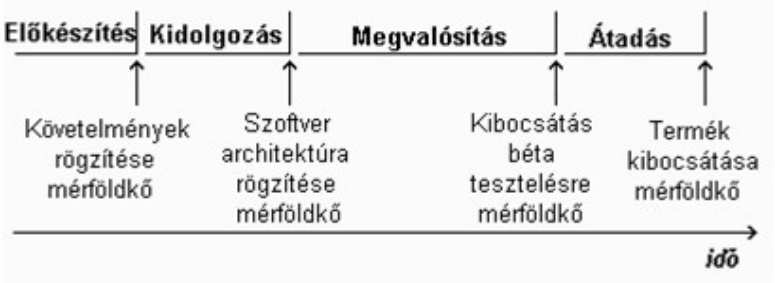
- **Komponensalapú** a szoftvert egyes komponensekből építi fel, amelyek külön-külön zárt egységeket képeznek, és egymással a megfelelő interfészeken keresztül kommunikálnak. Ennek köszönhetően a rendszer funkcionalitása különféle komponensek hozzáadásával könnyen alakítható, a megrendelő igényeihez igazítható.
- **Modellszemléletű** a rendszert különféle modelleken (elsősorban UML) keresztülközelíti meg, ezáltal igyekszik elérni a megfelelő funkcionalitást.
- **Use case vezérelt** a fejlesztés középpontjában a megrendelővel egyeztetett use case-ek (használati esetek) állnak, ezek pontos felmérése, majd megvalósítása elengedhetetlen a projekt sikeréhez.
- **Architektúra-centrikus** Kiemelt hangsúlyt kap a rendszer architektúrája, a megoldás felépítése, az egységbezárás és a laza csatolás általi felépítés.
- **Iteratív és inkrementális** a RUP életciklus modellje:



3-23. ábra

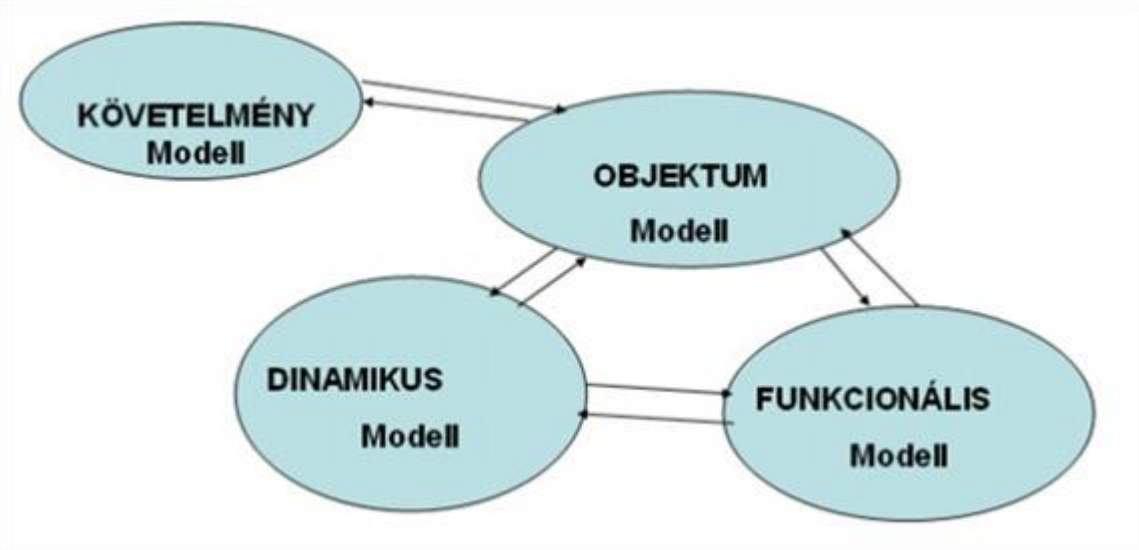
A Rational Unified Process a szoftverfejlesztés életciklusát négy egymást követő fázisra bontja: előkészítés (Inception), Kidolgozás (Elaboration), Megvalósítás (Construction), Átadás (Transition).

Minden fázis végén jól-definiált mérföldkövek vannak:



3-24. ábra

Az RUP módszertan a rendszert különböző nézőpontból felvett, összefüggő modellel szemlélteti.



Követelmény modell

A rendszer viselkedését, funkcionalitását írja le a szereplők és a feladatok megjelölésével, a felhasználó szemszögéből nézve. (A *szereplő* (actor) olyan személy vagy elem, amely kapcsolatban áll a rendszerrel, és aktívan kommunikál azzal, funkciókat indít el, vagy hajt végre.) A használati esetek jól meghatározott funkciók, amelyek végrehajtása üzenetváltást kíván. Meghatározó szerepet játszanak a fejlesztési folyamatban, hiszen a működés leírása a többi nézetet is jelentősen befolyásolja.

Objektum modell a rendszerben szereplő objektumokat írja le, attribútumaikat, műveleteiket és más objektumokkal való kapcsolataikat.

Dinamikus modell a rendszernek az idővel és a műveletek sorrendiségével kapcsolatos oldalát írja le. Azaz a változással járó eseményeket, események sorozatát, az események és állapotok elrendezését. A dinamikus modell a vezérlést rögzíti, a műveletek sorrendjét írja le tekintet nélkül arra, hogy azok mit tesznek, mit befolyásolnak, és hogyan valósulnak meg.

Funkcionális modell a rendszer adatainak átalakulását öleli fel: a függvényeket, a megfeleltetéseket, a kényszerfeltételeket és a funkcionális összefüggéseket. A funkcionális modell azt írja le, hogy mit tesz a rendszer, azt nem, hogy hogyan és mikor.

6. 3.6 Összefoglalás

Objektumorientált megközelítés a rendszerépítés olyan egységeit találta meg, amelyek a műszaki berendezések szabványos alkatrészeihez hasonlóan rugalmasan kombinálhatók, újrafelhasználhatók, lecserélhetők, amikből szilárd rendszer építhető, mert teljesen magukba rejtik a rendszer többi részére nem tartozó részleteket, elkülönítik az egymástól független komponenseket, ezzel megakadályozzák a tervet is érintő változások hatásának szétterjedését a rendszerben. A modellszemléletű RUP módszertan különböző modelljeinek részletes ismertetése a következő két modul anyaga.

Kérdések:

- Az információrendszerek fejlesztésének időrendi modelljei?
- Mi az életciklus és milyen szakaszokra bontható?
- Milyen életciklus-modelleket ismer? Miben különböznek ezek?
- Miért van szükség módszerekre, módszertanokra a rendszerek fejlesztésében?
- Mi a strukturált szemléletmód lényege, ismertesse az SSADM módszertan jellemzőit!
- A rendszerfejlesztés életciklusának mely szakaszaira terjed ki és melyekre nem az SSADM?

- Ismertesse az objektumorientált szemléletmód lényegét és az Objektumorientált módszertan jellegzetességeit!
- Mit jelent, hogy a RUP modellszemléletű?
- A rendszerfejlesztés életciklusának mely szakaszaira terjed ki, és melyekre nem a RUP módszertan?

Irodalomjegyzék

Raffai M.: *Információrendszerek fejlesztése és menedzselése,* Budapest, Novadat Bt., 2003.

I. Sommerville,: *Szoftverrendszerek fejlesztése,* Panem, Budapest, 2002.

Angster E.: *Az objektumorientált tervezés és programozás alapjai,* Angster E., Budapest, 1999.

http://www.sze.hu/~raffai/org/informatika1-2.pdf