1. tétel

Programozási és programtervezési technológiák fejlődése. Procedurális elvű technológiák. Fázisok, módszerek, ajánlások és szabványok. Eszközök és programozási környezet. Tesztelés. A RUP alapelvei.

Rendszertervezési megközelítések:

1. Funkcionális programozás:

A funkcionális programozásban minden entitás *függvény*. A program legfőbb jellemzője, hogy nincsenek benne *változók* - ez nagyon megkönnyíti az illető programozási nyelven írt programok helyességének a bizonyítását illetve megkönnyíti azt, hogy modulokból bizonyítottan helyes programokat *rakjunk össze*. Az, hogy a változók nincsenek, azt jelenti, hogy egy *"változó"* sem változtatja értékét: minden változó, ha egyszer értéket rendeltünk hozzá, akkor az életciklusa alatt az értéke nem változik.

2. Adatközpontú:

Középpontban az adatok állnak. → adatszerkezetekből indulunk ki
Cél: redundancia megszüntetése és az adatok egységesítése.
Középpontba került az adatmodellezés
A folyamatokat a műveletekig lebontjuk

3. Objektum-orientált

Adat, folyamat és viselkedés egyetlen objektumba sűrítve.
Nem a műveletek megalkotása áll a középpontban, hanem az egymással kapcsolatban álló programegységek hierarchiájának megtervezése →A tárgymodellt az "objektumok" strukturálják

A szoftverfejlesztési folyamat modelljei:
Szoftvergyártás

Tevékenységek olyan strukturált sorozata, amelyek a szoftverek kifejlesztéséhez szükségesek

- Specifikáció;
- Tervezés;
- Ellenőrzés (validáció);
- Továbbfejlesztés (evolúció).

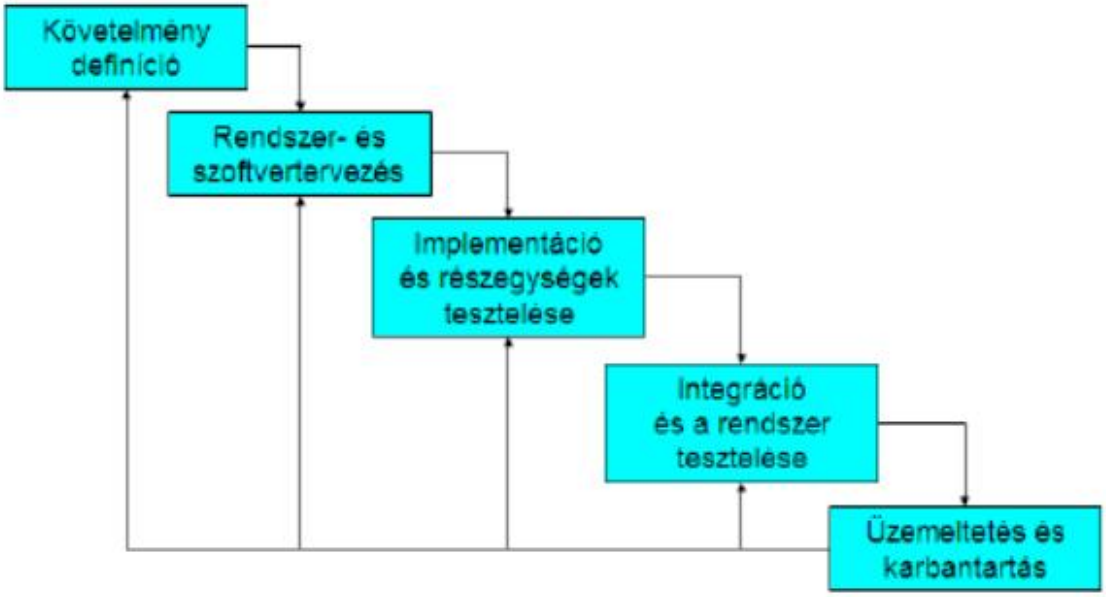
A szoftvergyártás absztrakt modellje a gyártási tevékenységet írja le egy adott nézőpontból.

Alapvető szoftvergyártási modellek

- A vízesés (waterfall) modell
  - Élesen elkülönülő specifikációs és fejlesztési fázisok.
- Evolúciós fejlesztési modellek
  - A specifikáció, fejlesztés és validáció átlapolódik.
- Komponens alapú fejlesztés
  - A rendszert kész komponensekből állítjuk össze.

A fenti modelleknek számos variációja létezik. Pl. formális fejlesztési modell: ez vízesés modellen alapul, ahol a specifikáció formális, ami sok lépésben finomítva elvezet az implementálható tervig.

A vízesés modell



A vízesés modell legfőbb hátrányai:

- A gyártás megindulás a után nehéz változásokat beépíteni.
- Egy munkafázisnak be kell fejeződni, mielőtt a következő elkezdődhet.

A vízesés modell problémái

Nehéz a változó megrendelői igényekhez igazodni, mert a projekt nehezen változtatható részegységekből áll.
Ez a modell akkor hasznos, ha a követelmények jól ismertek és csak nagyon kis változások lehetségesek a fejlesztés során.
Sajnos csak kevés üzleti rendszernek vannak stabil követelményei.
A vízesés modellt főleg nagy rendszerek fejlesztése során használják, ahol a fejlesztés több helyszínen történik.

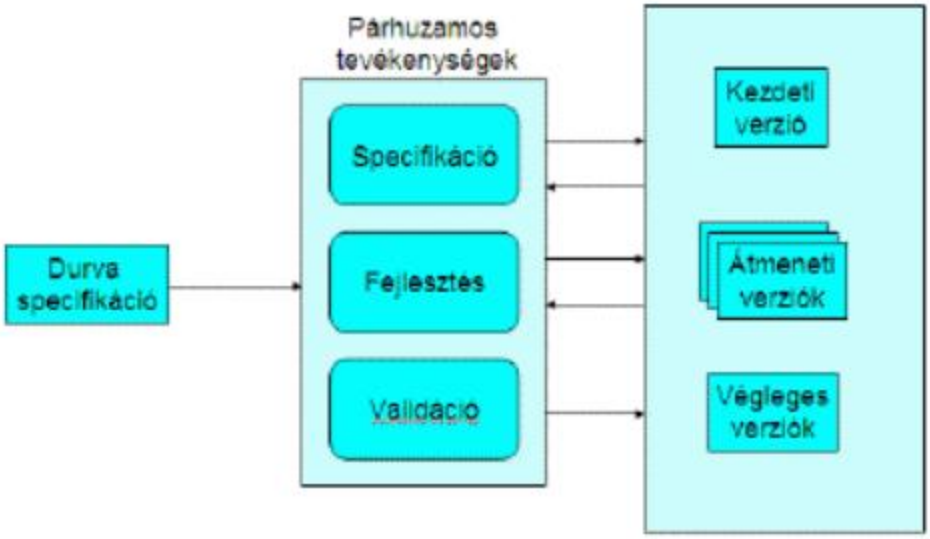
<u>Evolúciós fejlesztés</u>

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.



Evolúciós fejlesztés

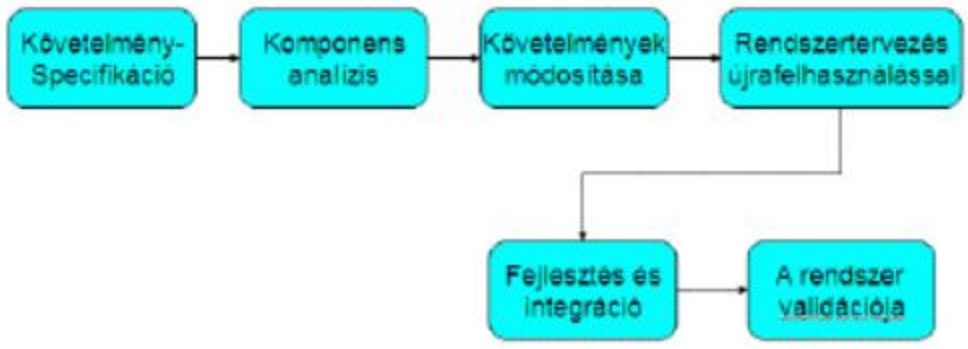
Problémák
- A fejlesztés nem átlátható;
- A rendszerek gyakran rosszul strukturáltak;
- Speciális felkészültségre lehet szükség (pl. rapid prototyping nyelvek).

Alkalmazhatóság
- Kis- és középméretű interaktív rendszerek;
- Nagy rendszerek részegységei (pl. felhasználói felület);
- Rövid élettartamú rendszerek.

Komponens-alapú szoftverfejlesztés

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.

Újrafelhasználás-alapú fejlesztés



Iteratív szoftvergyártás

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.
Két kapcsolódó megközelítés:
• Inkrementális teljesítés
• Spirális fejlesztés

Inkrementális teljesítés

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

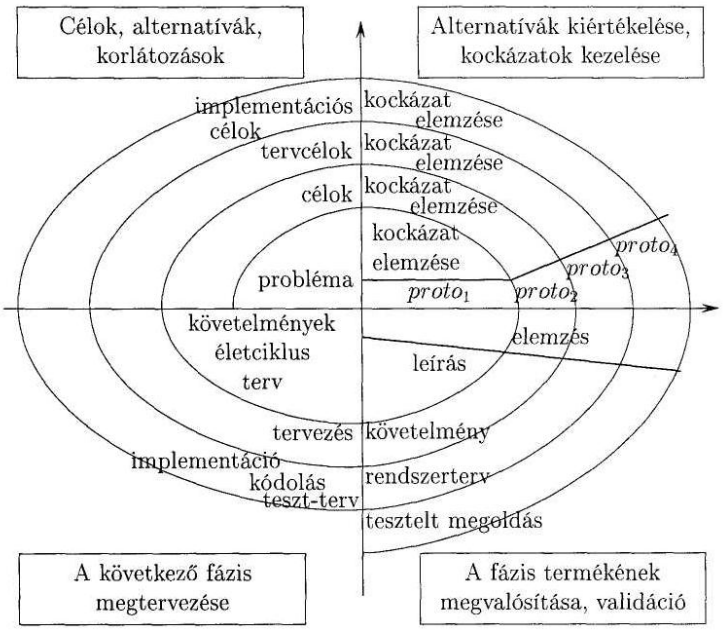
Minden átadás során működő részegységeket helyezünk üzembe. A rendszer korábban kezdheti meg (rész)működését.

Korábbi komponensek prototípusként működnek, így a későbbi részegységek követelménytervezésében ezek is segítenek.
Kisebb a projekt teljes csődjének esélye.
A legfontosabb szolgáltatásokat tesztelik a legtovább.

Spirális fejlesztés

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 szoftvergyártás spirális modellje



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

Rational Unified Process

Korszerű tervezési modell, amely az UML, és a hozzá kapcsolódó eljárásokból jött létre.
Általában három 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.

Az RUP legfontosabb újdonságai a fázisok és a munkafolyamatok szétválasztása, és az, hogy a felhasználói környezetben való üzembe helyezés a folyamat része. A fázisok dinamikusak és konkrét célokkal rendelkeznek, a munkafolyamatok statikusak és olyan technikai tevékenységeket tartalmaznak, amik a teljes fejlesztésen keresztülhúzódnak és nem rendelhetők hozzá az egyes fázisokhoz.

A szoftverfejlesztés fázisai:

1. **Követelményelemzés**

A követelmények leírásának tartalmaznia kell a válaszokat a következő kérdésekre:
- • Mi a probléma?
- • Mik a korlátozó tényezők? (Hardver, szoftver stb.)
- Mi a probléma elfogadható megoldása? (Minőség)

A követelmények elemzése során a következő kérdéseket kell megvizsgálni:
- Önmagában jó-e a követelmények leírása?
  - Nincs-e benne ellentmondás? *(Konzisztencia vizsgálat.)*
  - Teljes-e a leírás? *(Komplettség vizsgálat.)*
- A követelmények leírása megfelel-e a felhasználó által elképzelt problémának? (*Validáció vizsgálat.)*
- A követelményeket kielégítő megoldás megvalósítható-e? *{Megvalósíthatósági vizsgálat.)*
- A követelmények úgy vannak-e megfogalmazva, hogy azok tesztelhetők? *(Tesztelhetőség vizsgálata.)*
- A követelmények nem mondanak-e ellent a módosíthatóság, a továbbfejleszthetőség követelményének? *(Nyíltság kritériumainak vizsgálata.)*

2. **Elemzés**

A programspecifikáció a következő kérdésekre kell, hogy válaszoljon a követelmények leírása alapján:
• Mik a bemenő adatok? (Forma, jelentés, megjelenés.)
• Mik az eredmények? (Forma, jelentés, megjelenés.)
• Mi a reláció a bemenő adatok és az eredmény adatok között, mi azok viszonya?

- Már az elkészítendő szoftvert tervezzük, de megvalósítási részletek még nem érdekesek. A szoftver és a tárgyterület ideális modelljét keressük.
- A követelményelemzés alapján létrehozzuk a „fogalmi szótárt”
- A fogalmi szótár alapján létrehozzuk a tárgymodellt (domain modell)

- A funkcionális felosztás (modell) alapján létrehozzuk a szoftver működési tervét
- finomítjuk és kiegészítjük a tesztesetkatalógust

3. **Tervezés**
   - Az elkészítendő szoftvert tervezésénél minden fontos technikai részletet figyelembe veszünk.
   - Az elemzési modellből kiindulva meghatározzuk a szoftver-csomagok határait és összekapcsolódási tervét. Külön figyelemmel vagyunk a már meglevő csomagokra
   - Elkészítjük az újonnan kifejlesztendő szoftver csomagok modultervét, és a felületeik (interface) tervét
   - Az összekapcsolódási tervet a hardver elemekkel összehangolva elkészítjük a telepítési tervet
   - finomítjuk és kiegészítjük a tesztesetkatalógust
4. **Megvalósítás**
   - Végső tervezési döntések meghozatala
   - Kódolási szabványok összegyűjtése és dokumentálása (style guide)
   - Csoportmunka környezet létrehozása
   - Forráskódok létrehozása
   - Tesztszkriptek/kódok létrehozása
   - A szoftvert a forrásanyagból újrageneráló szkriptek létrehozása
   - Kezdeti adatállományt létrehozó (adatbázis populáció) szkriptek létrehozása
   - Terjesztő és telepítőszkriptek létrehozása
5. **Tesztelés**
   - Funkcióteszt (egységteszt, unit teszt): az egyes eljárások/csomagok hibamentes működését ellenőrzi
   - Integrációs teszt: az egyes szoftverösszetevők helyes működését, ill. a helyes összekapcsolódását ellenőrzi
   - Rendszerteszt: a teljes rendszert a végfelhasználó szemszögéből teszteli
   - Átvételi teszt: rendszerteszt, amelyet a szoftver átvételekor a megrendelő hajt végre
   - Regressziós teszt: integrációs teszt, változtatások után az érintett csomagok automatizált ellenőrzésére.
6. **Követés:**
   - Hibajavítás
   - Kismérvű módosítások megtétele
   - Frissítés a felhasználóknál (javítócsomagok)