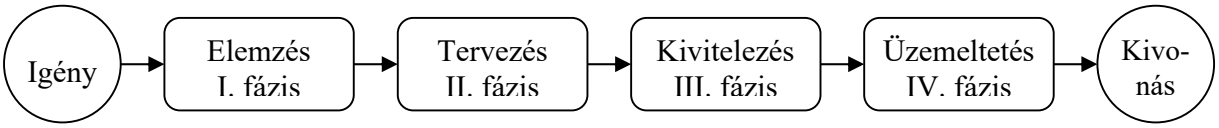
Szoftver életciklusmodellek[1]

Minden terméknek, így a szoftvereknek is van **életciklusa** (élettörténete), amely egy új termék (vagy egy meglévő módosítása) iránti igény felmerülésétől a használatból történő kivonásáig (selejtezésig) tart. Egy termék életciklusa a következőképpen modellezhető:



A fejlesztés életciklusa

A fázisok tartalma:

- I. fázis – elemzés, specifikáció: igények felmérése, környezet specifikálása, a szükséges erőforrások meghatározása, korlátok definiálása, megvalósíthatósági elemzés.
- II. fázis – tervezés: rendszerterv, kidolgozása, fejlesztési módszertanok kidolgozása, nagyvonalú (rendszerszintű) tervezés.
- III. fázis – kivitelezés: tervek végrehajtása, kódolás, programrészek összekapcsolása, tesztelés.
- IV. fázis - üzemeltetés: szoftver átadás, üzembe helyezés, üzemeltetés, terméktámogatás, a garanciális javítás, szerviz, kapcsolattartás a felhasználókkal.

Fejlesztésen egy termék életciklusának az első szakaszát értjük, amely az igény felmerülésétől a gyártás beindításáig tart.

A fejlesztőmunka hatékonyságának növelése, valamint a termék (szoftver) használhatóbbá, a célnak tökéletesebben megfelelővé tétele érdekében különböző fejlesztési elveket dolgoztak ki. Ezek figyelembe veszik a használhatósági, az illeszthetőségi, a módosíthatósági, a fejlesztési idő és az élettartam kritériumokat.

Napjainkig több fejlesztési folyamatmodellt dolgoztak ki, amelyek alapvetően abban különböznek egymástól, hogy az egyes fejlesztési fázisokat mennyire tartják fontosnak, megengednek-e átfedéseket, illetve visszacsatolásokat. A legelterjedtebben alkalmazott folyamatmodellek az alábbiak:

- életciklus-modellek
  - vízesésmodell,
  - visszacsatolásos vízesésmodell,
  - V-modell,
- Prototípusmodellek
  - gyors prototípus
  - evolúciós prototípus
- inkrementálismodell
- spirálmodell

[1] A tananyag Raffai Mária (2003): Információrendszerek fejlesztése és menedzselése (Novadat kiadó) felhasználásával készült

Szoftverek élettörténete

A szoftverek (programok) életciklusa jelentősen eltér az egyéb termékekétől, mivel a gyártás (a program előállítása) viszonylag egyszerű, viszont maga a termék bonyolult, jelentős tervezési (szellemi) ráfordítást igényel. Ennek megfelelően a modellek elsősorban nem a gyártásra, hanem a termék fejlesztésére koncentrálnak.

További eltérés, hogy a szoftverek technikailag könnyen módosíthatók, akár a gyártás után is. Ennek az lehet a következménye, hogy azonos gyártmányok (kész termékek) életciklusa eltérhet egymástól.

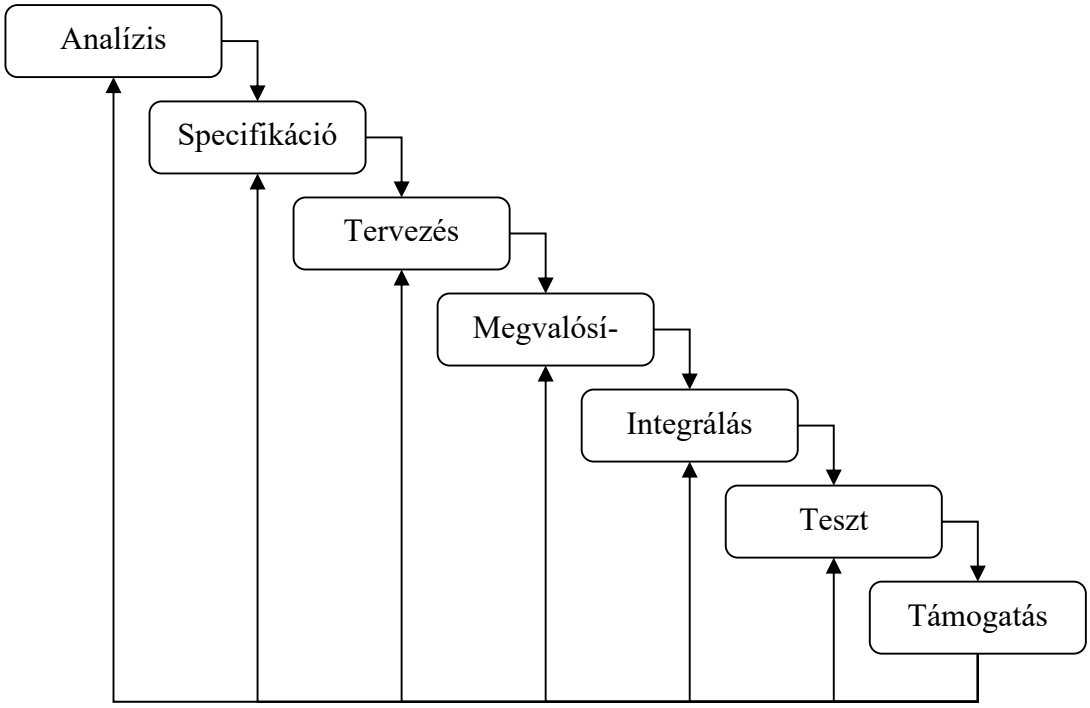
A szoftverfejlesztés során „nulláról" kiindulva egy meghatározott igényt kielégítő szoftver első példányát hozzuk létre, vagy egy meglévő szoftver továbbfejlesztését végezzük el. A szoftverek általában sokkal összetettebbek, bonyolultabbak, mint a hagyományos (anyagi természetű) termékek, ezért a „nulláról" induló fejlesztések esetén nem minden esetben törekednek a teljes, minden igényt kielégítő példány előállítására, hanem azokat folyamatosan fejlesztik.

Amennyire (viszonylag) egyszerű a módosítások végrehajtása, annyira óvatosságra intetnek a bonyolultságból adódó veszélyek. Ugyanis előfordulhat, hogy nemcsak a kívánt célt érjük el, hanem a módosítások hatással lehetnek más programrészekre is.

Életciklus-modellek

Vízesésmodell - Waterfall Model

A szoftverfejlesztés életciklusát leíró modell a vízesésmodell, amelyet fázismodellnek is neveznek. Nevét a szemléltető ábra jellegzetes alakjáról kapta:



A vízesésmodell

A vízesésmodell ábrája jól szemlélteti az egyes tervezési fázisokat és a tevékenységek egymásutániságát.

A fejlesztési folyamat I. (elemzés) fázisa az analízis, amely magába foglalja a feladat definiálását, és a megvalósíthatósági elemzést is.

A II. (tervezési) fázis tartalmazza a követelményspecifikációt, a projectindítást, a fogalmi, architektúrális és fizikai (részletes) tervezést.

A III. (kivitelezés) fázisnál átfedést találunk a tervezési szinten a II. fázissal. Ide soroljuk a rendszertervezés egyes elemeit, a megvalósítást (programkódolást, implementációt), integrálást (programrészek összekapcsolását, operációsrendszerhez történő illesztését) és a tesztelést is.

A IV. (üzemeltetés) fázisban a fejlesztő biztosítja a terméktámogatást, a garanciális javításokat, szervizt, és a felhasználókkal való kapcsolattartást.

Az ábrán látható, hogy a fejlesztés csak meghatározott sorrendben (szekvenciálisan) történhet, és csak akkor léphetünk a következő elemre, ha az adott feladatot lezártuk (mérföldkő). Ez legtöbbször az adott szinthez tartozó dokumentáció elkészítésével zárul. A továbblépéshez szükséges, hogy a felhasználó elfogadja az előző fázis eredményeit.

A modellnél nincs lehetőségünk a visszalépésre.

A modell abban az esetben lehet hatékony, ha a felhasználó pontosan tudja, hogy mit akar, és azt definiálni is képes. Ellenkező esetben jelentősen megemelkedhet a fejlesztés költsége.

A modellnél látható, hogy az első fázis (elemzés) eredménye nagymértékben kihat a további munkára és a termék használhatóságára. Az elemzés során viszont olyan személyekkel (felhasználó, megbízó stb.) kell együtt dolgozni és kialakítani a koncepciót, akik nem jártasak a fejlesztésben. Abban érdekeltek, hogy a megszokott (ismert) módszert, formátumot alkalmazzák számítógép segítségével és idegenkednek minden ismeretlentől. Legtöbbször a felhasználó nem is tudja felmérni teljes mértékben az igényeit. Gyakran előfordul, hogy a késztermék láttán jutnak eszébe új dolgok, vagy a felismerés: „ezt másképpen kellett volna csinálni”.

A szoftverfejlesztés folyamata legtöbbször nem egy lineáris modell, hanem a fejlesztési tevékenységek iterációjának sorozata. Mivel e modellnél minden egyes javítás (módosítás) a folyamat újraindításával jár, ezért az iterációk költségesek.

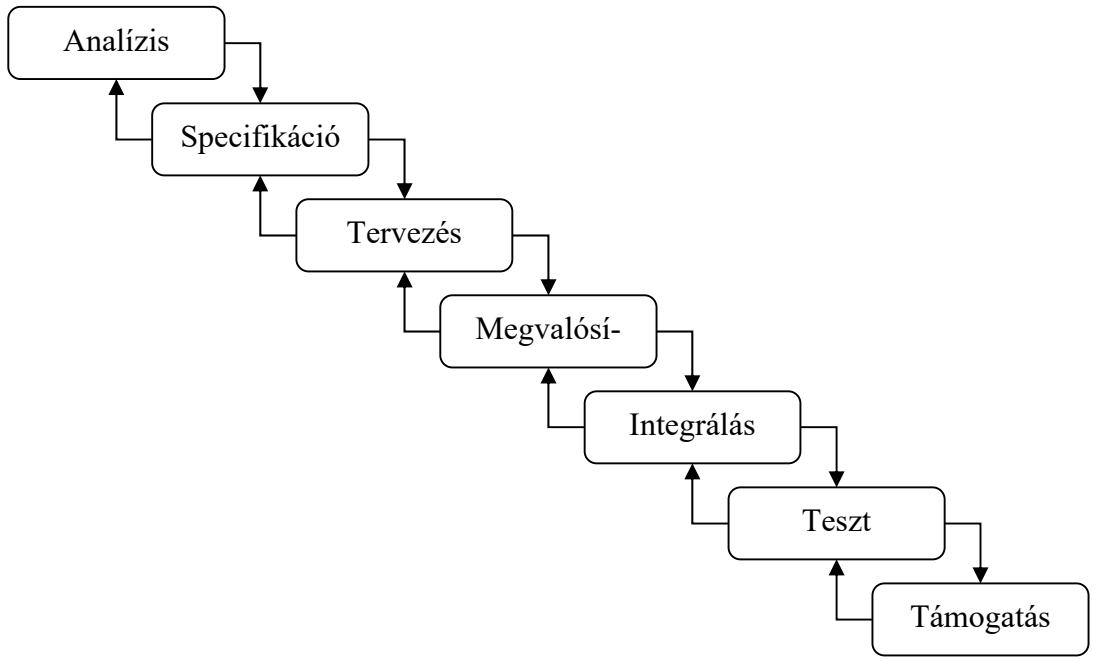
A vízesésmodell megalkotása jelentős lépés volt a szoftverfejlesztés területén, ugyanis e modell megalkotásával vált világossá, hogy a szoftverfejlesztés nem egyenlő a programírással.

Manapság a modellt– elsősorban korlátai miatt – kevés helyen alkalmazzák. Elsősorban a nagyon kicsi rendszereknél, ahol akár egy ember is képes átlátni az egész folyamatot, vagy a nagy, egyedi rendszerek fejlesztésének átfogó leírásánál. A nagy rendszerek részfeladatainak fejlesztésére azonban már más módszert (modellt) alkalmaznak.

A modell korlátai akkor jelentkeznek, amikor a tesztelések vagy az átadás során kiderül – a leggondosabb tervezés ellenére – valamilyen módosításra van szükség. A szoftver-felhasználókra jellemző tulajdonság, hogy gyakran igénylik a változtatást.

Visszacsatolásos vízesésmodell

Egy program fejlesztése során gyakran előfordul, hogy valamelyik fázisban rájövünk arra, hogy az előzőekben hibáztunk vagy a lefektetett elképzelések, tervek nem jók, azok módosításra szorulnak. Tehát szükségünk van olyan modellre, amely lehetőséget ad egy korábbi fázisba történő visszalépésre. Ez a visszacsatolásos vízesésmodell:

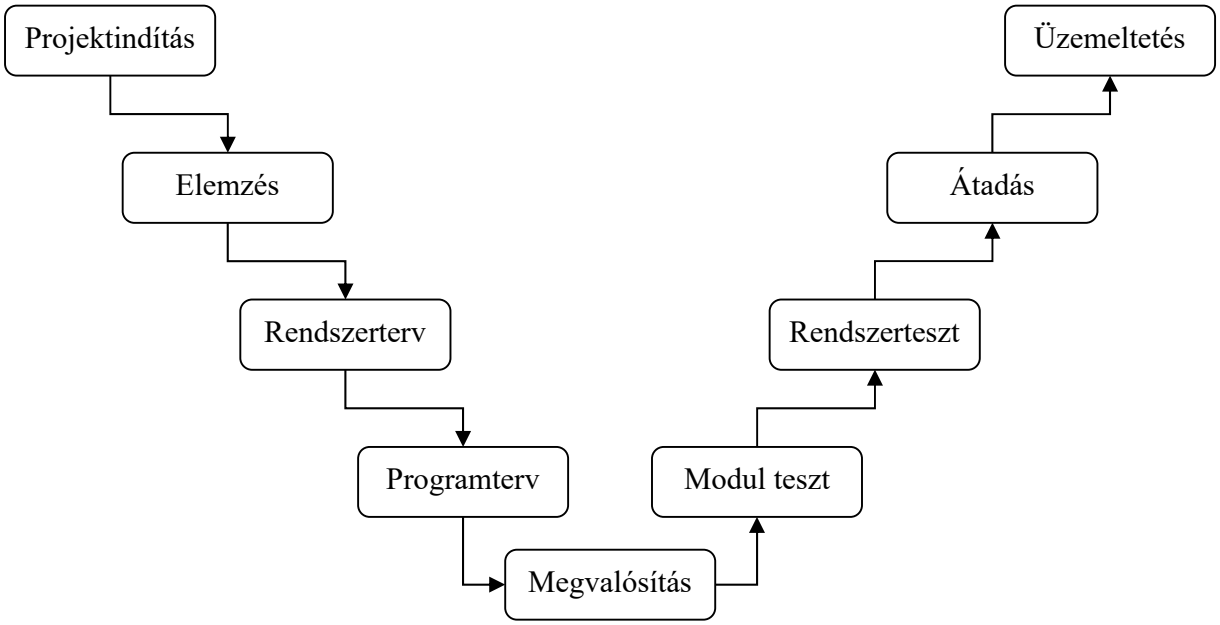


A visszacsatolásos vízesésmodell

Az ábrán látható megoldással elkerülhetők a komolyabb, a fejlesztési életciklus teljes hosszában végigmenő hibák. Egy hiba javításának költsége függ attól, hogy hány fázison keresztül kell visszalépnünk.

V-modell

A vízesésmodell egy újabb, továbbfejlesztett változata a V-modell, amely a fejlesztés első két fázisát a leszálló ágba, a megvalósítást a modell aljára, az implementációt, tesztelést, a modell bevezetését a felszálló ágára helyezi:



A V-modell

A modellnél minden tervezési fázis után a fejlesztés két részre válik: egyrészt megkezdődik a következő fázis végrehajtása, másrészt megtervezésre kerül az adott fázis tesztrendszere.

V-modell fázisai:

1. projektindítás, célkitűzések, definíciók, tanulmányok elkészítése,
2. elemzés, rendszerkövetelmények meghatározása, megvalósíthatósági elemzés,
3. rendszerterv, nagyvonalú architektúrális rendszerterv elkészítése,
4. részletes alrendszer, modulterv, programterv elkészítése,
5. megvalósítás, kódolás (programozás), programmodulok elkésztése,
6. modul- és integrációs tesztek,
7. rendszertesztek,
8. átadás, üzembe helyezés, átvételi tesztek,
9. üzemeltetés, rendszerfelügyelet, karbantartás, aktualizálás.

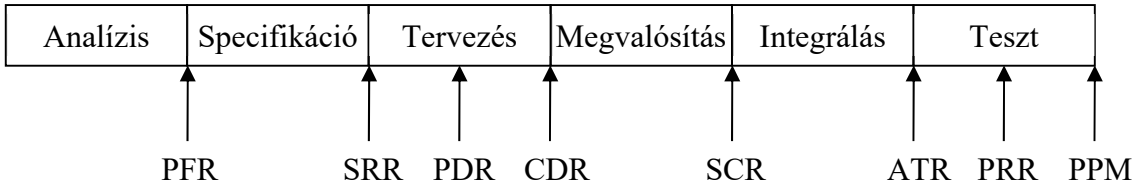
A V-modell leszálló ágában lévő fázisokat logikailag a tesztfázisok kapcsolják össze az azonos szinten lévő felszálló ági fázisokkal. Ennek megfelelően a rendszertesztek a 2. és 8., az integrációs tesztek a 3. és 7., a modultesztek a 4. és 6. fázisok között biztosítják a logikai kapcsolatot.

Az életciklus-modellek összefoglalása

Az előzőekből látható, hogy az életciklus-modellek egymástól jól elkülönülő, egymást lineárisan követő diszkrét tevékenységek sorozatából épül fel. Az egyes tevékenységek csak akkor kezdődhetnek el, ha az előtte lévő már befejeződött. Ebből következik, hogy előre meg kell határoznunk azokat a kritériumokat (mérföldköveket) is, amelyek alapján el tudjuk dönteni, hogy egy adott fázis mikor fejeződik be. A fázisok határainak meghatározására ajánlásokat és a szabványokat dolgoztak ki, amelyek célja nemcsak a dokumentáció elkészítése és

átvizsgálása, hanem annak ellenőrzése is, hogy az algoritmus megfelelően működik-e (verifikáció) valamint, hogy a specifikáció mennyire illeszkedik az elvárásokhoz (validitás).

Az egyik ilyen szabvány, az USA Védelmi Minisztériuma által kidolgozott DoD-2167 számú szabvány, amely szigorú előírásokat tartalmaz a készítendő dokumentációra vonatkozóan. A fejlesztési modellt, és a modulokhoz rendelt mérföldköveket az ábra szemlélteti:



DoD-2167 által megszabott dokumentációk

Az egyes fázisokhoz rendelt mérföldkövek kódjainak magyarázata:

PFR – *Product Feasibility Review*	Megvalósíthatósági vizsgálat
SRR – *Software Requirements Review*	Rendszerkövetelmények analízise
PDR – *Preliminary Design Review*	Előzetes, nagyvonalú tervek áttekintése
CDR – *Critical Design Review*	Részletes tervek áttekintése
SCR – *Source Code Review*	Forráskódok felülvizsgálata
ATR – *Acceptance Test Review*	Elfogadhatósági teszt vizsgálata
PRR – *Product Release Review*	Forgalomba hozatal előtti áttekintés
PPM – *Project Post-Mortem Review*	Projekt kiértékelése

Az életciklus modellekkel történt fejlesztés eredménye egy működő számítógépes rendszer, amelyet üzemeltetni kell. Mivel e modellek elsősorban a fejlesztés, kivitelezés folyamatát adják, így általában nem foglalkoznak az üzemeltetési, rendszerkövetési fázissal, hogy a rendszer átadás után hogyan működik, mennyire felel meg a felhasználó elvárásainak.

Az életciklus-modell előnyei:

- világos a folyamat struktúrája,
- egyszerű megvalósítás,
- a tervezési fázis feladatainak egységes szemléletű végrehajtása.

Hátrányok:

- a valóságos folyamatok ritkán követik ezt a szekvenciális modellt,
- az iterációs lépések és az előre nem látott hibák javítása magas költséggel járnak,
- a módszer feltételezi az igények pontos megfogalmazását,
- a felhasználó csak a tervezési folyamat végén fog működőképes programot látni.

A gyakorlatban a felhasználóknak (megrendelőknek) csak elképzelésük van a fejlesztés céljáról, de nem tudják pontosan definiálni a rendszer paramétereit, a leprogramozni kívánt valóságos folyamatokat, a rendszerrel szemben támasztott követelményeket és az elvárt eredményeket. E problémák kiküszöbölésére újabb folyamatmodelleket alkottak meg.

Prototípusmodell

Prototípusszemléletű fejlesztésnek nevezzük azt a módszert, amikor egy – a valóságos működést szimuláló – modellt készítünk, azzal a céllal, hogy a felhasználó könnyebben el tudja dönteni, hogy mire van szüksége. A tapasztalatok azt mutatják, hogy egy fejlesztési folyamat bizonytalanságáért nagymértékben a megrendelő (felhasználó) a felelős. Például nem tudja pontosan definiálni az elvárásait, vagy nem tudja azokat egzakt, érthető módon közölni a fejlesztővel. A bizonytalanságok elkerülésének legjobb módja, ha be tudjuk mutatni a fejleszteni kívánt rendszer prototípusát.

A prototípus készítésének éppen az a célja, hogy a felhasználó nagyvonalú (kezdeti) elképzelései alapján készítsünk el egy olyan működő modellt, amely képes a felhasználóval történő kommunikációra. Így a kezdeti eredmények elemzése alapján a felhasználó (megrendelő) pontosítani tudja elképzelését, amelyeket a fejlesztő beépít a prototípusba. Ennek érdekében először a felhasználói felületek (többször grafikus felületek) készülnek el.

Prototípus kifejlesztésénél már figyelembe vehetjük a fejlesztéssel járó kockázatokat is. Ha minimális kockázatra (költségre) törekszünk, akkor célszerű a prototípust papíron (képernyőtervek, ábrák, rajzok) elkészíteni. Nagyobb kockázatot jelent, ha a kész alkalmazás szimulációjára vállalkozunk egyszerű programok segítségével. A legnagyobb kockázatot akkor vállaljuk, ha a teljesen kifejlesztjük a rendszert.

A prototípusmodell használata akkor hatékony, ha a fejlesztő rendelkezik a prototípus gyors előállításához szükséges erőforrásokkal.

Prototípusok készítésére többféle lehetőség adott:

- magas szintű fejlesztőeszköz (programkörnyezet) alkalmazása, ahol fontos a feladathoz legjobban illeszkedő technika és programozási nyelv kiválasztása,
- negyedik generációs (4GL) nyelvek használata, amelye természetesen adatbázis-kezelésre is alkalmasak,
- meglévő (újrafelhasználható) elemek felhasználása.

A prototípusmodell előnyei:

- a megrendelővel való szoros együttműködés miatt kicsi az esélye annak, hogy a rendszer nem elégíti ki a felhasználó elvárásait, illetve több felesleges funkció épül be a rendszerbe,
- a megrendelő már a fejlesztés kezdetén ismerkedik a termékkel, így elvárásait folyamatosan beépíthetjük a rendszerbe, és nemcsak a végén, mint a vízesésmodell esetében,
- nem szükséges szigorúan betartani a fejlesztés kötött sorrendjét, egyes részeket külön-külön is elkészíthetünk, ami az egyes elemek párhuzamos fejlesztését teszi lehetővé,
- a szoftver legfontosabb elemei (interfészek, input és output modulok), a fejlesztés korai stádiumában megvitathatók a megrendelővel.

Hátrányok:

- a megrendelő egy működő rendszer lát, de többnyire nincs tisztában azzal, hogy ez csak a fejlesztés kisebb hányada, ugyanis a prototípusból legtöbbször hiányoznak azok az elemek (adatellenőrzés, rendezés, keresés stb.), amelyek elsősorban meghatározzák egy végleges rendszer paramétereit: fejlesztési költség, futási és várakozási idő, illetve adatellenőrzés, adatbiztonság,

- mivel a fejlesztés I. (elemzés) fázisa nem annyira teljes, mint az előző modellek esetében, menet közben derülhet ki, hogy rossz kiindulást választottunk, vagyis a választott programkörnyezettel (programnyelv, adatbázis kezelő stb.) nem teljesíthetők a végső elvárások,
- hiányzik a teljes életciklusra kialakított egységes koncepció, így a részrendszerek külön fejlesztéséből adódhatnak kompatibilitási, együttműködési problémák, azaz a rendszer nem áll össze egésszé.

A prototípus modelleket (előnyük és hátrányuk miatt) gyakran ötvözik más (például vízesés) modellel, ilyenkor a nehezen érthető vagy kisebb részrendszereket fejlesztik e módszerrel.

Gyors prototípusmodell - Rapid Prototyping Model

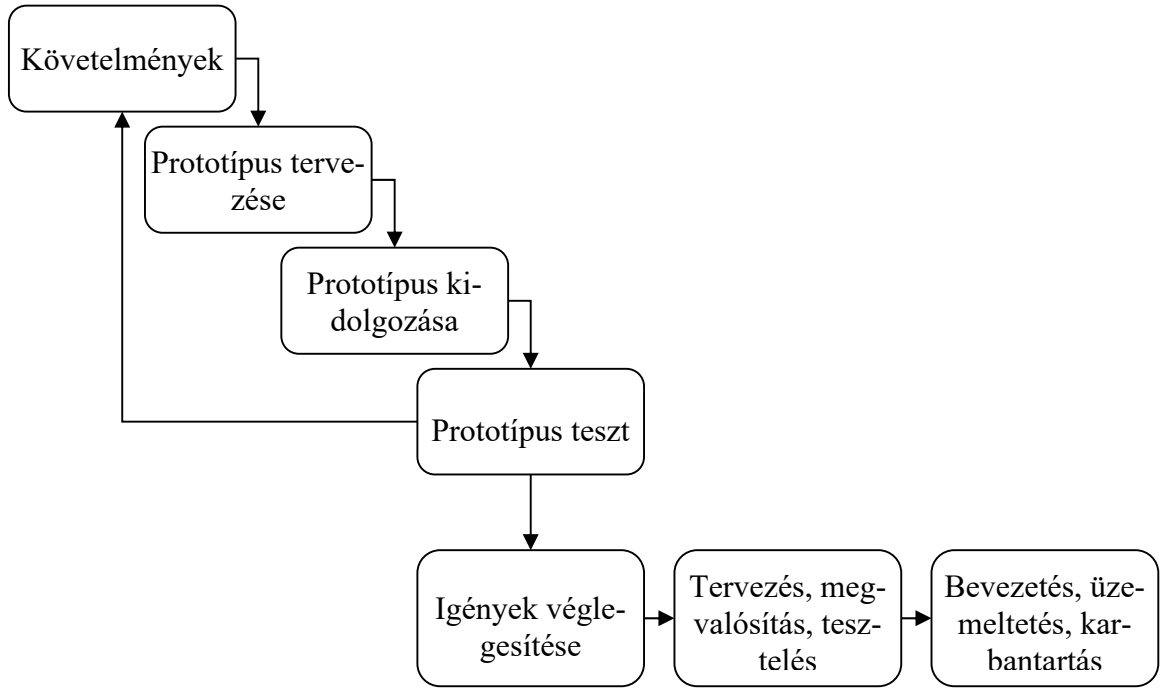
A gyors prototípus modell hasonlít a vízesésmodellre, azzal a különbséggel, hogy az első fázis végrehajtása során az aktuális ismeret alapján elkészítjük a termék prototípusát.

E módszer csak akkor lehet hatékony, ha a folyamatban résztvevők (elemzők, tervezők, programozók) rendelkeznek a gyors prototípus előállításához szükséges ismeretekkel.

A modell előnye, hogy az elvárásokat könnyebb meghatározni, már a fejlesztés korai szakaszában lehetőség van az analízisre, a felhasználó aktívan közreműködhet a fejlesztés prototípus fázisában. Hátránya, hogy a felhasználónak jelentős többletfeladatot ad, valamit a prototípus kidolgozásának elhúzódása kihat a teljes projekt befejezésének időpontjára.

Evolúciós prototípusmodell

Ennél a modellnél a prototípusfejlesztés iteratív módon történik, vagyis az első prototípus elkészítése után azt a felhasználó elemzi, teszteli, majd észrevételeit, javaslatait beépíthetjük a prototípusba. A fejlesztés (iteráció) addig tart, amíg a megrendelő elvárásai és a megvalósítás közötti különbség egy elfogadható szint alá csökken, elkészül a végleges változat. (100%-os készültségű szoftvert nem lehet kifejleszteni, mert a felhasználónak mindig lesznek újabb és újabb ötletei.)



Evolúciós prototípusmodell

Az evolúciós prototípusmodellnek két csoportját különböztetjük meg:

1. a feltáró (explorációs) fejlesztés, ahol a kezdeti specifikáció felállítása, a követelmények feltárása és a végleges prototípus kialakítása a megrendelővel együtt történik. Ezáltal könnyebben megérthetők a követelmények, valamint a megrendelő által felvetett módosítások, esetleg új funkciók, tulajdonságok hozzáadhatók a prototípushoz.
2. eldobható prototípusok fejlesztése esetén a követelmények ismeretében több prototípusváltozat kerül kifejlesztésre, amelyeket a megrendelő kipróbálhat és így alakítja ki a véleményét.

Az evolúciós fejlesztés jellemzői:

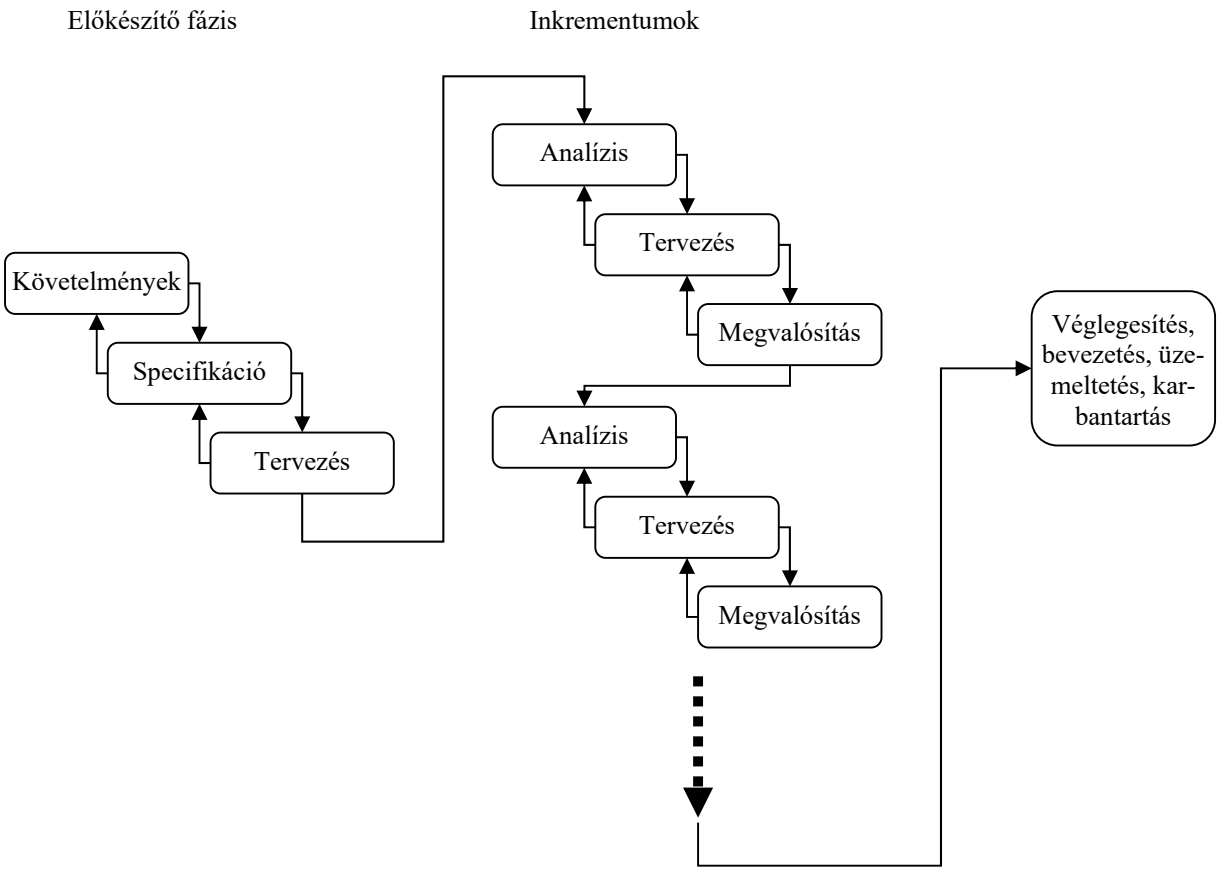
- felhasználó-orientált fejlesztés, vagyis maximálisan figyelembe veszi a fejlesztő kívánságait,
- a folyamatos változtatások miatt a folyamat nehezen átlátható, rosszul strukturált, a haladás nehezen mérhető,
- a gyors fejlesztéshez megfelelő szaktudás és speciális eszközök (erőforrások) szükségesek,
- rövid életciklusú, kis és közepes méretű, illetve interaktív rendszerek fejlesztésére alkalmas,
- a nagy szoftverrendszerek esetén annak részeinek fejlesztésére használják.

Inkrementális modell

A különböző fejlesztési modellek megalkotásával az volt a cél, hogy minimálisra csökkentsék a fejlesztés kockázatát, valamint az elért eredmények és az elvárások közötti különbséget. Már a prototípusmodellek is alkalmasak voltak arra, hogy eltérjünk az életciklus-modellek által megkövetelt szigorú sorrendtől. Ha a fejlesztés során a kritikus komponenseket készítjük el előbb, és azokat lépésről-lépésre, folyamatos visszacsatolással fejlesztjük, akkor inkrementális fejlesztési modellről beszélünk.

Mint a következő ábrából is látszik, a fejlesztést – az előkészítő fázis során – egy kezdeti modell kidolgozásával kezdjük, majd ezt a gyakorlati használhatóság és a megrendelő követelményeinek megfelelően több lépésben javítjuk. Így az egyes részfeladatok több, kipróbálásra alkalmas verziója, *inkrementuma* készül el.

Az inkrementális fejlesztés esetén nincs szigorúan előírt sorrend, így lehetőségünk van – a fejlesztés kockázatának csökkentése érdekében – arra, hogy a rendszer problematikus részeit fejlesszük ki először. Ha megfelelnek, akkor ezeket kiegészíthetjük az újabb és újabb részekkel. Vagyis a fejlesztési folyamatban – szemben a vízesésmodellel – a rendszert nem egyenletesen fejlesztjük teljes szélességében, hanem bizonyos (problematikus) modulokat előre dolgozunk ki. A két modell természetesen kombinálható. Mint az ábrán is látható egyes inkrementum fejlesztése a vízesésmodell alapján történhet.



Inkrementális modell

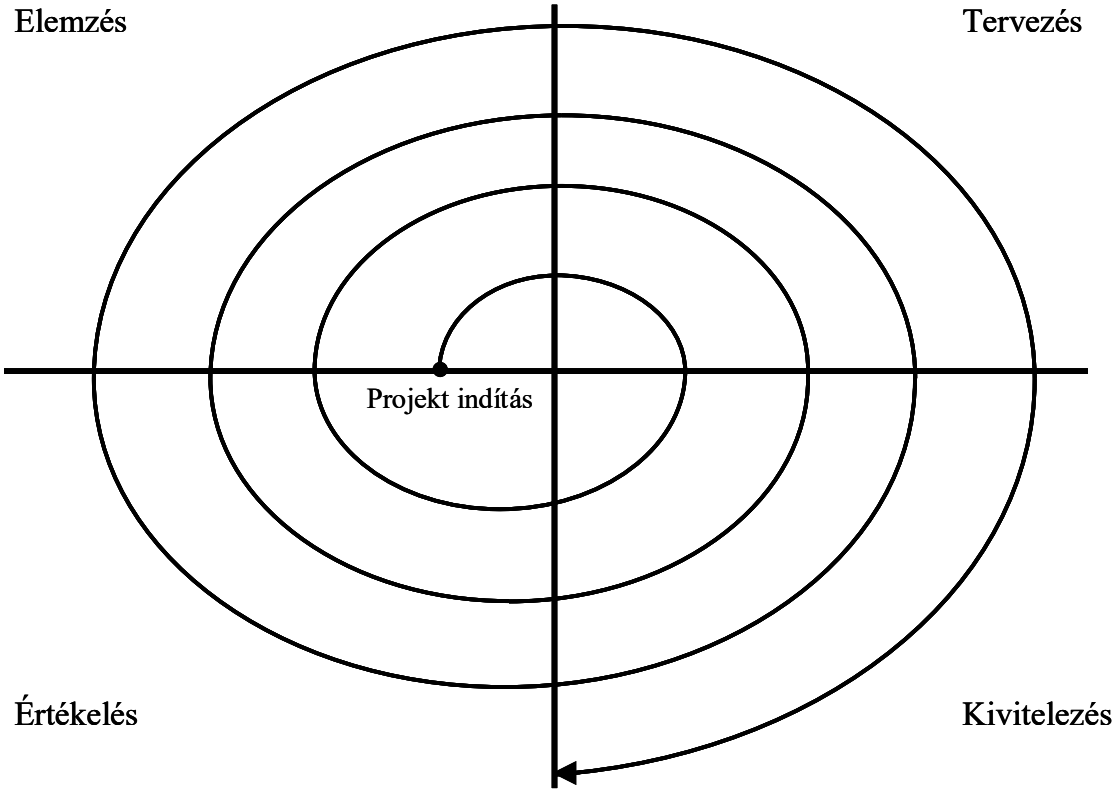
Az inkrementálismodell előnye, hogy a módszer akkor is használható:

- ha előre nem definiált pontosan a rendszer felépítése, algoritmusai, a megoldhatóság kérdései, illetve a szükséges erőforrások,
- a felhasználó nem tudja pontosan meghatározni a fejlesztéssel vagy a rendszerrel kapcsolatos elvárásait, illetve azokat nem tudja egzakt módon megfogalmazni.

Hátránya, hogy a hosszú átfutási időt és soklépéses fejlesztést igényel.

Spirálmodell

A fejlesztési modelleket és azok tulajdonságait elemezve C.W. Boehm[2] egy olyan modellt javasolt, amely a kockázatminimalizálásra törekszik. A spirálmodellnek nevezett fejlesztési folyamatot az ábra szemlélteti:



Spirálmodell

Mint az ábrából is látszik, az életciklus a ponttal jelölt helyről indul, ahol a követelmények és a kockázatok meghatározása történik (elemzés térrész), majd további három térrészen át körbe fut. A második térrészben a logikai és fizikai tervezésen kívül gazdasági szemléletű kockázatelemzés is megvalósul. (A lehetséges alternatívák közül mindig a minimális kockázattal rendelkezőt választjuk.) A harmadik térrészben a választott alternatíva megvalósítása (programtervek kidolgozása, programfejlesztés, kódolás, tesztelés), majd az utolsó térrészben az elemzés, értékelés, és ha szükséges, a következő fázis tervezése történik meg. Az itt szerzett tapasztalatok alapján dől el, hogy szükséges-e módosítás, továbbfejlesztés, vagyis rátérünk-e a következő körre (iterációs ciklusra). A folyamatnak (spirálnak) akkor van vége, ha a termék kielégíti a felhasználó igényeit. Ezután megtörténhet a szoftver átadása, üzembe helyezése és üzemeltetése.

A spirál mentén haladva: az origótól mért távolság (sugárirányú dimenzió) a halmozott költségeket, a szögelfordulás az egyes iterációkra fordított időt szemlélteti.

A spirálmodell elsősorban elméleti modell, gyakorlatban a különböző fejlesztési modellek kombinációja, például a fejlesztés különböző fázisai más-más modellel is megvalósíthatók. A modell előtérbe helyezi a ciklusonkénti prototípus-építést,

[2] Boehm, B.W. 1986: A Spiral Model of Software Development, ACM SIGSOFT, SE Notes