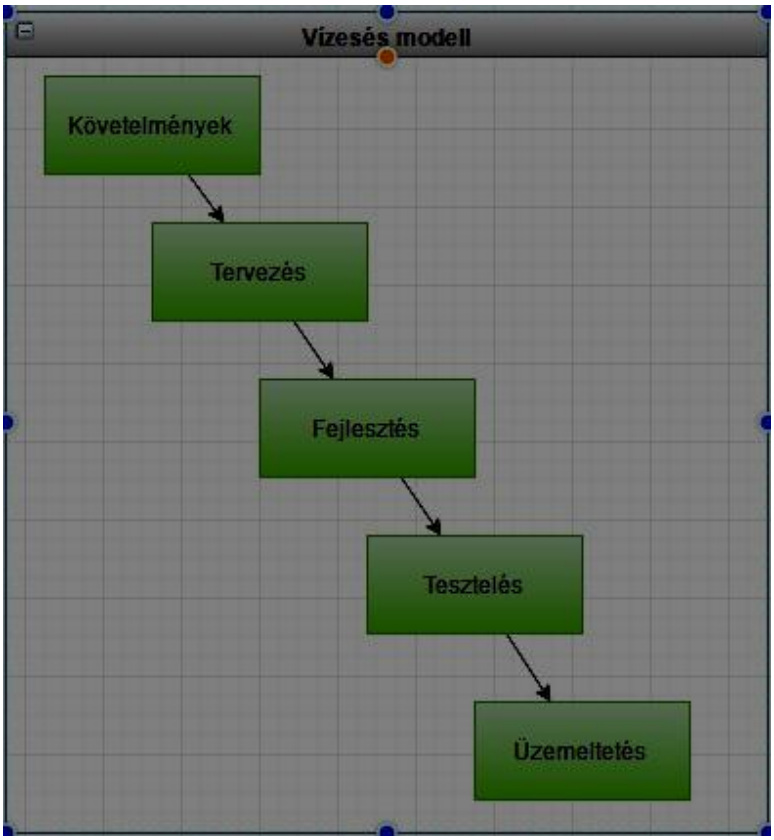
A fejlesztési módszertanok evolúciója

Az elmúlt 20-30 évben több fejlesztési módszertan is született, amelyek az adott kor igényeinek próbáltak megfelelni, vagy éppen egy speciális terület speciális igényeit próbálták lefedni. A technikai fejlődés és a felgyorsult életvitel is megköveteli, hogy a módszertanokat időről-időre felülvizsgálják, és módosítsák, vagy újat találjanak ki.

Vízesés modell



A legelső modell, mindenhol ezzel kezdik a módszertanok bemutatását. 1970-ben mutatták be, jellemzője, hogy az egyes tevékenységeket szeparálja, és ezeket lépésekben, egymással átfedésben, mintegy vízesésszerűen mutatja be. Az eredeti specifikáció szerint az egyes lépések addig nem kezdődhetnek el, amíg az előző be nem fejeződött, de a gyakorlatba ez már úgy került át, hogy a következő lépés már elkezdődik akkor, amikor az előző már a befejezéséhez közeledik. Fontos, hogy minden lépésben keletkeznek dokumentumok, amik a következő lépés bemenő dokumentumai, azaz a következő lépés ezen a dokumentumokon alapszik.

A lépések a következők:

Követelményspecifikáció. A hardver és szoftver oldali igények felmérése.

Követelmények elemzése: Az előző lépésben leírt követelmények feldolgozása, modellek készítése.

Tervezés: Itt történik meg a leendő szoftver megtervezése, az architektúra kialakítása, a modulok, funkciók definiálása, a függőségek, adatmodellek leírása.

Fejlesztés: Az előző lépésben keletkezett rendszertervnek megfelelően elkészül a szoftver.

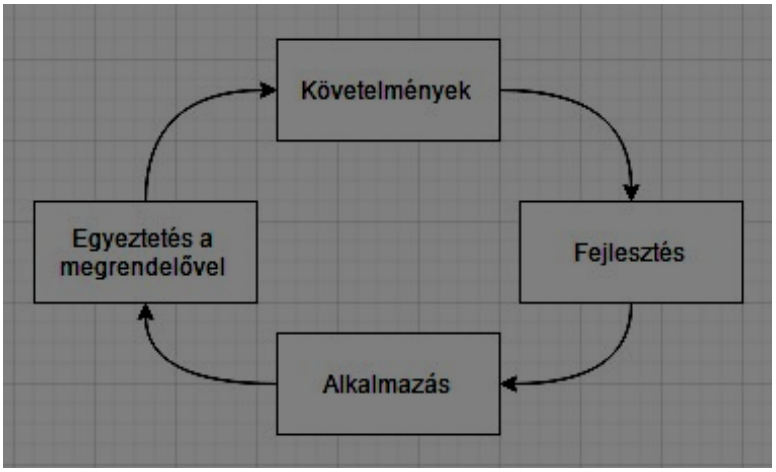
Tesztelés: Az elkészült szoftvert tesztelni kell, funkció, modul és rendszerszinten is.

Átadás, üzemeltetés, karbantartás: Az elkészült, letesztelt szoftver átadják, beüzemelik, az ekkor felmerülő hibákat javítják, az esetleges igényeket lefejlesztik. Az új igények lefejlesztése önmagában lehet egy kisebb vízesés modell alapja.

Bár már lassan 50 éves modell, de még napjainkban is használják, főleg olyan alkalmazások fejlesztésében, ahol a megrendelői igények jól definiálhatók, nem változnak, és garantálható, hogy a fejlesztés ideje alatt nem történik az igényekben módosítás. Ez leginkább nagyvállalati környezetben jellemző, ahol a működést 5-10 éves intervallumra előre tervezik meg.

Evolúciós modell

Ez a fajta megközelítés azon alapszik, hogy a kész szoftver nem egy ciklusban készül el, hanem több, egymást követő ciklusban folyamatosan finomítják az igényeket, és igazítják hozzá a programot. Az első lépésben a fejlesztők elkészítenek egy alapot, amit véleményeztetnek a megrendelővel, ennek alapján azt módosítják, újra véleményeztetik, újra módosítanak. Minden iterációnak az alapja az előző iteráció. Az iterációk száma nincs előre meghatározva, a folyamat akkor ér véget, amikor a megrendelő képviselői számára megfelelő a program. Ez a megközelítés jobban alkalmazható abban az esetben, ha a megrendelő még nem teljesen bizonyos az igényeiben, nincs minden információ birtokában, így lehetőséget az a munka megkezdésére már sokkal korábbi szakaszban, mint a vízesés modell. Ezáltal dinamikusabb tud lenni, a megrendelő több visszajelzést kap, nagyobb a ráhatása és



rálátása a fejlesztésre, biztosabb lehet abban, hogy azt kapja, amit megrendelt.

Az evolúciós fejlesztésben kétféle típust különböztetünk meg:

Felderítő vagy feltáró fejlesztés: Itt az a cél, hogy az egyes iterációkon át egyre jobban közelítsük az igényeket, és a végén egy működő alkalmazást kapjon a megrendelő.

Eldobható prototípus készítés: A célja, hogy az igényeket a lehető legjobban megértsük, és ehhez készítsük el a lehető legjobb követelményspecifikációt. A fő cél, hogy mindig a legkevésbé érthető részeit pontosítsuk a követelményeknek a következő iterációban.

Komponens alapú fejlesztési modell

Az modell alapja, hogy arra kell törekedni, hogy olyan modulokra bontsuk a programot, amit többször is fel lehet használni a fejlesztés során. Az ugyanis megszokott dolog, hogy egy összetettebb program sok olyan funkciót tartalmaz, amelyek nem csak egy folyamatban találhatók meg. Ezeket össze kell gyűjteni, és a kész komponenst felhasználva megírni a kódot. A modell 4 fő szakaszra bontható:

Komponenselemzés: Az igények felmérésekor meg kell keresni a lehetséges komponenseket, amik kiemelhetők a funkciók közül.

Követelménymódosítás: Ha megvan a lehetséges komponensek köre, akkor meg kell vizsgálni és adott esetben módosítani a követelményeket a kigyűjtött komponenseknek megfelelően. A módosítást követően újra komponenselemzést kell végezni.

Rendszertervezés újrafelhasználással: A lehetséges komponensek meghatározása után meg kell tervezni az alkalmazást a komponensekre alapozva, különös tekintettel azok újrafelhasználását tekintve.

Fejlesztés és integráció.

Ezen fejlesztési modell előnye, hogy a tervezés során csökkentik az egymást átfedő funkciók számát, logikusabbá téve az alkalmazás felépítését. Ezzel csökken a fejlesztési idő és költség, és több idő fordítódik a tervezésre. Hátránya, hogy a megrendelő oldaláról nagyobb kompromisszumkészséget igényel, mert a komponensek kialakítása legtöbbször az eredeti igények kisebb-nagyobb módosításával történik meg.

Iteratív fejlesztés

Gyakorlatilag a vízesés és az evolúciós modelleket ötvözi. A vízesés modellel ellentétben nem egy lineáris folyamatnak képzeli el a fejlesztést, hanem több, egymást követő tevékenységek iterációjaként. Ezek a ciklikus tevékenységek, iterációk mindig a rendszer változtatását vonják maguk után. Ezt a módszert az élet is igazolja, hiszen az igények az első definiálás után folyamatosan változnak, amik megkövetelik egy program folyamatos továbbfejlesztését. (Érdekesség, de az igények módosulásának intervalluma egyre rövidül, régebben jellemzően 5-10 év is volt, míg az igények változása miatt egy program funkcióinak 50%-a nem módosult, ez ma már jellemzően 1-2 év alá csökkent.) Ezen fejlesztési módszertannal a követelményspecifikáció és a szoftver folyamatosan fejlődik, lehetőséget adva a fejlesztés közben felmerülő változások követésére.

Két jellemző implementálása van:

Inkrementális fejlesztés: A követelmények specifikálása, a tervezés, programozás kis lépésekre van bontva.

Spirális fejlesztés: A folyamatban a program folyamatosan újabb funkciókkal bővül, mintegy spirálszerűen, a megrendelő folyamatosan egy egyre többet tudó programot kap.

Rational Unified Process

A Rational Unified Process a Rational Software Corporation cég fejlesztése, amely céget az IBM 2003-ban felvásárolta. Nem egy kész kidolgozott folyamat, inkább egy keretet ad, amit minden cég saját magára szabhat. A RUP a Unified Process módszertanra épül, azt egészíti ki. Valójában annak felismerése nyomán jött létre, hogy az addigi szoftverfejlesztési módszertanok csak erősen leegyszerűsítve illeszthetők a valós folyamatokra, ezért minden addigi modellből építkezik, kiemeli a legjobb részeket, és megpróbálja egy közös, jól működő rendszerbe illeszteni ezeket. Alapvetően iteratív fejlesztési stratégiát ír le.

A RUP módszertan kitalálásakor alapvetően 3 probléma minél hatékonyabb megoldását tűzték ki célul:

- A projekt erőforrás-igényének minél pontosabb becslését.

  Minden cégvezetés elsőrendű igénye, hogy a várható kiadásokkal a lehető leghamarabb tisztában legyen. Az iterációkra bontással becsülhetővé teszi az időigényt, amely az előrehaladással egyre pontosabbá válik.

- Pontos, jól definiált célkitűzéseket a fejlesztést tekintve.

  A költségek és a fejlesztési idő mindig a legszűkebb keresztmetszet a fejlesztésben. Ezért mérföldköveket kell meghatározni, az egyes mérföldköveket pedig pontosan kell definiálni, hogy adott mérföldkövön belül mit és mikorra kell megvalósítani. Minden egyes elkészült mérföldkő végén meg lehet vizsgálni az addigi eredményeket, és a tapasztalatokat felhasználni a következő mérföldkövekben.

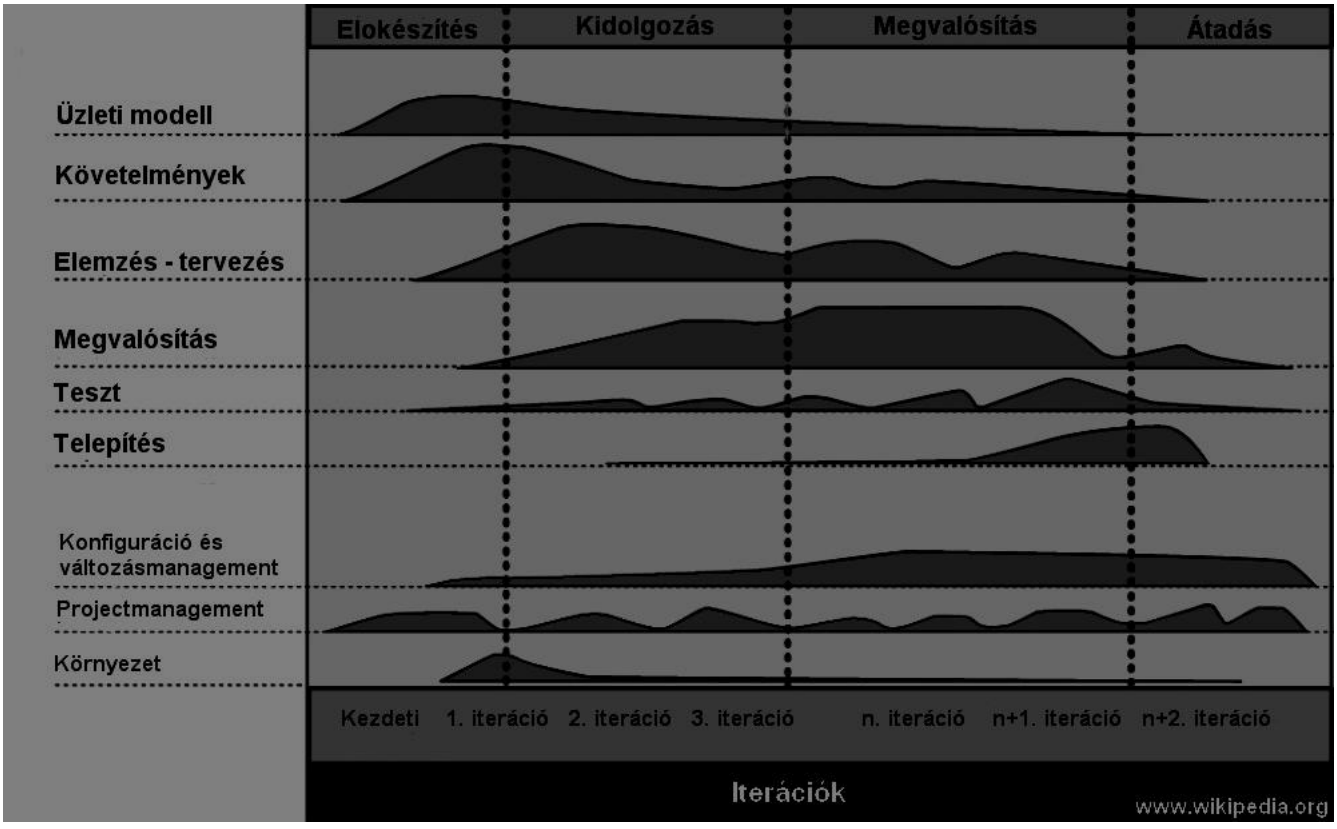
- A hibák okozta károk csökkentését

  Mivel a RUP iteratív módszertan, ezért a hibákat meg kell próbálni az adott iterációban megkeresni, és kijavítani, mert minél később kerül erre sor, annál nagyobb erőforrásokat fog igényelni a javítás.

A RUP elemekből, úgynevezett „building block”-okból áll. Ezek a következők:

- Szerepkör (ki)
- Tevékenység (hogyan)
- Termékek (mit)

A Szerepkör írja le a szükséges képességeket, kompetenciákat, felelősségeket. A tevékenység egy munkafolyamat, aminek a végén a Termék(ek)et állítja elő a Szerepkörrel rendelkező személy.



Az iterációban a tevékenységet 6+3 kategóriára bontják.

Mérnöki tevékenységek:

- Üzleti modell
- Követelmények
- Elemzés -tervezés
- Implementáció
- Teszt
- Telepítés

Támogató tevékenységek:

- Konfiguráció és változásmanagement
- Projectmanagement
- Környezet

Egy project négy részből áll:

- Előkészítés
- Kidolgozás
- Megvalósítás
- Átadás

Az **Előkészítés** fázisban az eredeti ötletet át kell dolgozni, meg kell határozni, hogy milyen felhasználók milyen módon használhatják a rendszert, ki kell találni az architektúrát, az adatszerkezetet.

A **Kidolgozás** fázisban megtörténik a részletes tervezés, funkciókra bontás, jogosultságok meghatározása, a részletes, és stabil architektúra kialakítása. Az alap Unified Process metodika szerint az emberi test részei a csontváz, az izmok és a bőr. A részletes architektúra ebben a felfogásban a csontváz bőrrel fedve, minimális izomzattal.

A **Megvalósítás** fázisban készül el a rendszer (béta), ekkor kerül az izomzat a bőr alá.

Az **Átadás** fázisban történik meg a béta tesztelés, a felfedezett hibák javítása.

Az egyes részek állhatnak egy vagy több iterációból. Minden iteráció tartalmazza a mérnöki és a támogató tevékenységeket. Az egymást követő iterációk során az előző iteráció egy javított, bővített változatát kapjuk. Ezeket release-eknek, vagy kiadásoknak nevezzük.

A folyamat során többféle modell is készül(het):

- Üzleti elemzés
    - Business model (Üzleti modell)
- Követelmények
    - Use-case model (Használati eset modell)
- Elemzés
    - Analysis model (Elemzési modell)
- Tervezés
    - Design model (Tervezési modell)
    - Deployment model (Telepítési modell)
- Implementáció
    - Implementation model (Megvalósítási modell)
- Teszt
    - Test model (Teszt modell)

Az Üzleti modellben kell összeszedni a megrendelői igényeket, meghatározni az alapvető üzleti folyamatokat, fogalmakat, és a lehetséges felhasználókat. A követelmények meghatározása közben kell összegyűjteni az elvárt követelményeket, elképzeléseket. A követelmények részletezésénél ki kell fejteni a funkcionális és a nem-funkcionális követelményeket is. Ez utóbbiak például az alkalmazandó technológiák, szerverkörnyezet, bővíthetőség, tervezett terhelés. A követelmények meghatározását mindig a felhasználók szempontjából kell megtenni.

Az elemzési modellben írják le a követelményeket a fejlesztők számára. Így a követelményekben írják le a rendszer külső képét, míg az elemzési modellben a rendszer belső képét. E kettő dokumentum adja a későbbi rendszer fejlesztésének az alapját. Ilyenkor elemzik és rendszerezik az összegyűjtött használati eseteket, amelyből kialakítják a rendszer vázát. A tervezés folyamán ezt a vázat bővítik, egészítik ki teljessé úgy, hogy mind a funkcionális és a nem-funkcionális követelményeknek is eleget tegyenek. Így pontosan meg kell határozni a

technológiai hátteret, a rendszer modulokra bontását, ezek kapcsolódását, az interfészeket, protokollokat. A Unified Process ezt blueprintnek, tervrajznak hívja. Ez alapján teljesen egyértelműen meg kell tudni valósítani az adott alkalmazást.

Az implementációban elkészül a rendszer úgy, hogy közben az egyes önálló részek, modulok tesztelése meg is történik. Ugyanitt megtörténik a végleges architektúra kialakítása is. A teszt során megtervezik a teszteseteket és az iterációk végi rendszerteszteket, valamint a tesztesetek feldolgozása is itt történik meg.