Szoftverfolyamat modelljei

A szoftverfolyamat modellje a szoftverfolyamat absztrakt reprezentációja. Minden egyes modell különböző speciális perspektívából reprezentál egy folyamatot, de ily módon csak részleges információval szolgálhat magáról a folyamatról. Ezek az általános modellek nem a szoftverfolyamat pontos, végleges leírásai. Sokkalta inkább hasznos absztrakciók, amelyet a szoftverfejlesztés különböző megközelítési módjainak megértéséhez használhatunk.

Az irodalomban számos folyamatmodell alakult ki az évek során, melyek több éves tapasztalatokat gyűjtenek össze, és próbálják a szoftverfejlesztést, mint folyamatot egy magasabb szintre emelni. Külön kiemelendő, hogy az utóbbi évek modern Agilis irányzata (lásd később) új megközelítésekkel reagált a felgyorsuló szoftverfejlesztési igényekre. Ezekre a kurzus során még külön kitérünk. Most azonban a legismertebb, a klasszikus szoftverfejlesztési irányokat követő modelleket soroljuk fel, későbbiekben ezekre részletesen is kitérünk:

1. A vízesésmodell: Ez a folyamat alapvető tevékenységeit a folyamat különálló fázisainak tekinti. Ezek a fázisok a követelményspecifikáció, a szoftvertervezés, az implementáció, a tesztelés stb.

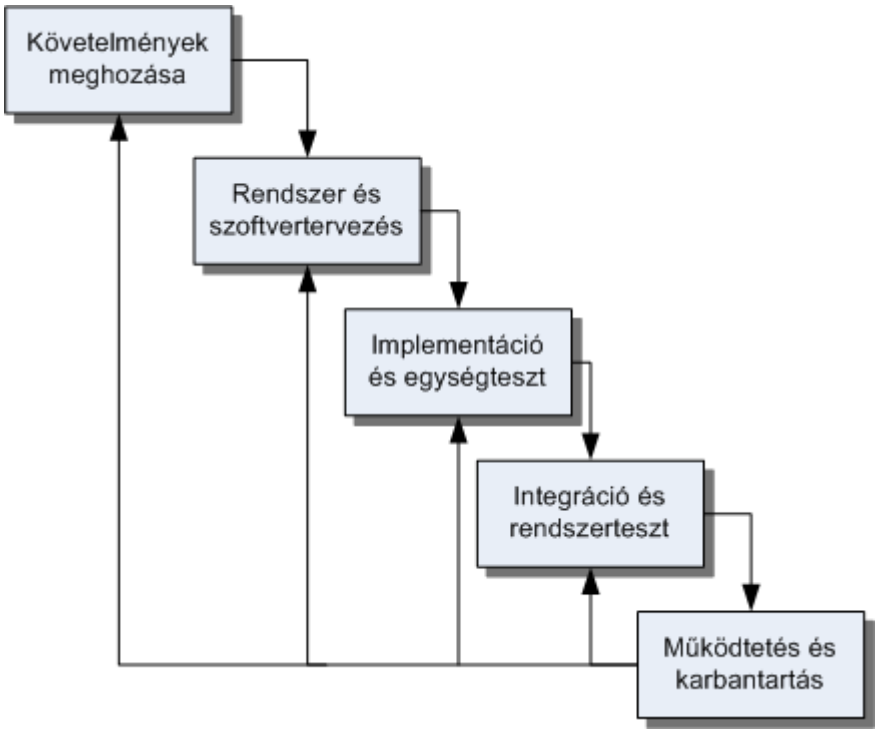
2. Evolúciós vagy iteratív fejlesztés: Ez a megközelítési mód összefésüli a specifikáció, a fejlesztés és a validáció tevékenységeit.

3. Komponens alapú fejlesztés: Ez a megközelítés nagy mennyiségű újrafelhasználható komponensek létezésén alapszik.

A gyakorlatban ezek a modelleknek gyakran keverednek is egymással, főként nagy rendszerek fejlesztésekor.

A Vízesés modell

A szoftverfejlesztés folyamatának első publikált modellje, amely más tervezői modellekből származik. Az elnevezése onnan fakad, hogy a modellben az egyes fázisok lépcsősen kapcsolódnak egymáshoz, ami alapján vízesésmodellként vált ismertté.



A fázisok eredménye tulajdonképpen egy dokumentum. A modell fontos szabálya, hogy a következő fázis addig nem indulhat el, amíg az előző fázis be nem fejeződött. A gyakorlatban persze ezek a szakaszok kissé átfedhetik egymást. Maga a szoftverfolyamat nem egyszerű lineáris modell, hanem a fejlesztési tevékenységek iterációjának sorozata. Ez a vízesésmodellnél a visszacsatolásokban jelenik meg.

1. **Követelmények elemzése és definiálása**: A rendszer szolgáltatásai, megszorításai és célja a rendszer felhasználóival történő konzultáció alapján alakul ki. Ezeket később részletesen kifejtik, és ezek szolgáltatják a rendszer specifikációt.

2. **Rendszer- és szoftvertervezé**s: A rendszer tervezési folyamatában választódnak szét a hardver- és szoftverkövetelmények. Itt kell kialakítani a rendszer átfogó architektúráját. A szoftver tervezése az alapvető szoftverrendszer-absztrakciók, illetve a közöttük levő kapcsolatok azonosítását és leírását is magában foglalja.

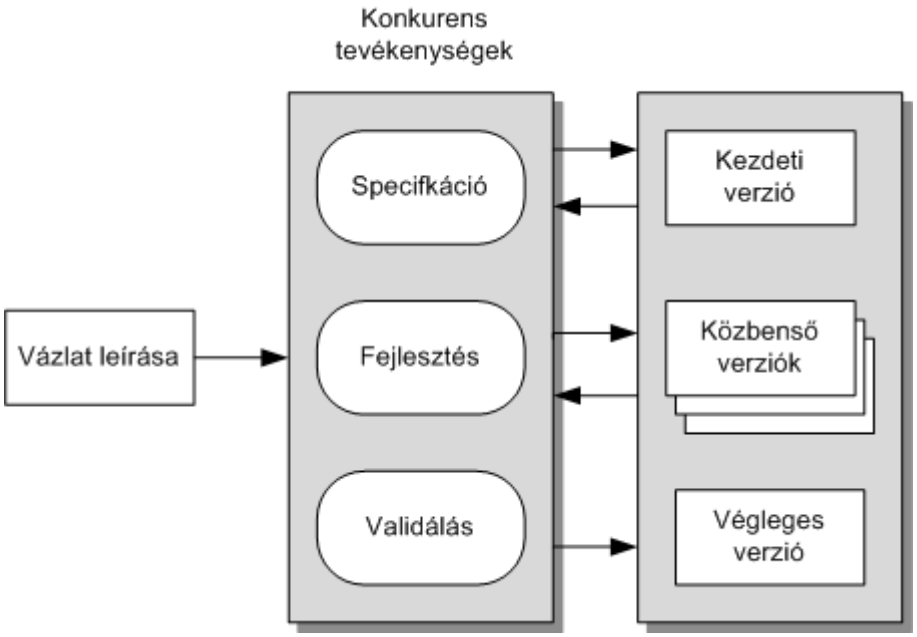
3. **Implementáció és egységteszt**: Ebben a szakaszban a szoftverterv programok, illetve programegységek halmazaként realizálódik. Az egységteszt azt ellenőrzi, hogy minden egység megfelel-e a specifikációjának.

4. **Integráció és rendszerteszt**: Megtörténik a különálló programegységek, illetve programok integrálása és teljes rendszerként való tesztelése, hogy a rendszer megfelel-e a követelményeknek. A tesztelés után a szoftverrendszer átadható az ügyfélnek.

5. **Működtetés és karbantartás**: Általában ez a szoftver életciklusának leghosszabb fázisa. Megtörtént a rendszertelepítés és megtörtént a rendszer gyakorlati használatbavétele. A karbantartásba beletartozik az olyan hibák javítása, amelyekre nem derült fény az életciklus korábbi szakaszaiban, a rendszeregységek implementációjának továbbfejlesztése, valamint a rendszer szolgáltatásainak továbbfejlesztése a felmerülő új követelményeknek megfelelően.

Evolúciós modell

Az evolúciós fejlesztés alapötlete az, hogy a fejlesztőcsapat kifejleszt egy kezdeti implementációt, majd azt a felhasználókkal véleményezteti, majd sok-sok verzión keresztül addig finomít, amíg a megfelelő rendszert el nem érik. A szétválasztott specifikációs, fejlesztési és validációs tevékenységekhez képest ez a megközelítési mód sokkal jobban érvényesíti a tevékenységek közötti párhuzamosságot és a gyors visszacsatolásokat.



Próbáljuk meg összevetni az evolúciós megközelítést a vízesésmodellel. A fentiek alapján láttuk, hogy a vízesésmodell kevésbé rugalmas a menetközben szükséges változásokra, így érvelhetünk azzal, hogy az evolúciós megközelítés hatékonyabb a vízesésmodellnél, ha olyan rendszert kell fejleszteni, amely közvetlenül megfelel az ügyfél kívánságainak. További előnye, hogy a rendszerspecifikáció inkrementálisan fejleszthető. Mindezek ellenére a vezetőség szemszögéből két probléma merülhet fel:

1. A folyamat nem látható: a menedzsereknek rendszeresen szükségük van leszállítható részeredményekre, hogy mérhessék a fejlődést.

2. A rendszerek gyakran szegényesen strukturáltak: a folyamatos változtatások lerontják a rendszer struktúráját, így kevésbé érthetővé válik. A szoftver változásainak összevonása pedig egyre nehezebbé és költségesebbé válhat.

Felmerülhet akkor a kérdés, hogy mikor és kinek érdemes használni az evolúciós fejlesztési modellt? Nos a válasz természetesen nem lehet egyértelmű, de a gyakorlati tapasztalatok alapján a várhatóan rövid élettartalmú kis vagy közepes rendszerek esetén célszerű az alkalmazása. Körülbelül 100.000 programsorig terjedően. Ugyanis nagy, hosszú élettartalmú rendszerek esetén az evolúciós fejlesztés válságossá válhat pontosan az evolúciós jellege miatt.

Evolúciós fejlesztés típusai

Az evolúciós fejlesztésnek két különböző típusa ismert:

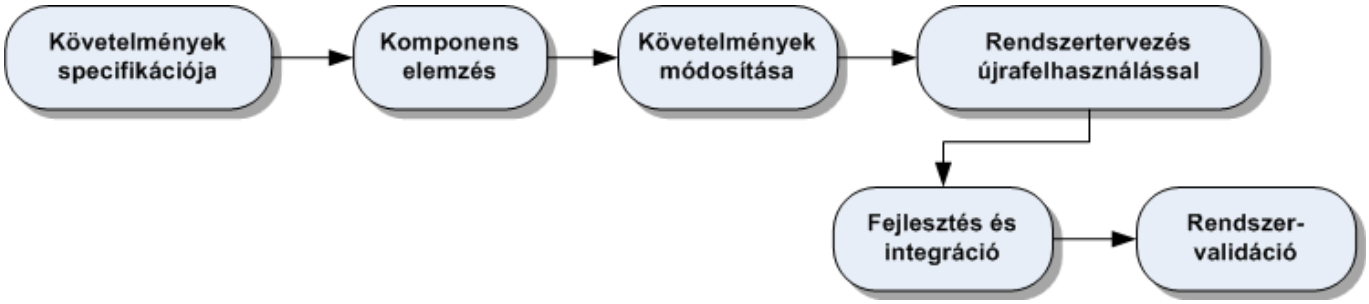
1. **Feltáró fejlesztés**: célja egy működőképes rendszer átadása a végfelhasználóknak. Ezért

elsősorban a legjobban megértett és előtérbe helyezett követelményekkel kezdik a fejlesztés menetét. Ennek érdekében a megrendelővel együtt tárjuk fel a követelményeket, és alakítják ki a végleges rendszert, amely úgy alakul ki, hogy egyre több, az ügyfél által kért tulajdonságot társítunk a már meglévőkhöz. A kevésbé fontos és körvonalazatlanabb követelmények akkor kerülnek megvalósításra, amikor a felhasználók kérik.

2. **Eldobható prototípus készítés**: A fejlesztés célja ekkor az, hogy a lehető legjobban megértsük az ügyfél követelményeit, amelyekre alapozva pontosan definiáljuk azokat. A prototípusnak pedig azon részekre kell koncentrálni, amelyek kevésbé érthetők.

Komponens alapú fejlesztés

A komponens alapú fejlesztés alapgondolata az, mint ahogy az elnevezés is utal rá, ahogy próbáljuk meg az elkészítendő szoftvert **újrafelhasználható** komponensekből felépíteni. Erre az ad okot, hogy a szoftverfolyamatok többségében megtalálható valamelyest a szoftverek **újrafelhasználása**. Ilyen esetekben előkeresik a korábbi kódot (komponenst) és újra átdolgozva, esetleg általánosítva beledolgozzák a rendszerbe.



Az újrafelhasználás-orientált megközelítési mód nagymértékben az elérhető újrafelhasználható szoftverkomponensekre, illetve azok egységes szerkezetbe történő integrációjára támaszkodik. Néha ezek a komponensek saját létjogosultsággal rendelkeznek. Amíg a kezdeti követelményspecifikációs és validációs szakasz összehasonlítható más folyamatokkal, addig a közöttük található szakaszok az újrafelhasználás-orientált fejlesztésekben különböznek.

Komponens alapú fejlesztés fázisai

1. **Komponenselemzés**: az adott követelményspecifikációnak megfelelően megkeressük azon komponenseket, amelyek megvalósítják azok funkcióit, implementálták azokat. A legtöbb esetben nincs egzakt illeszkedés, a felhasznált komponens a funkciók csak egy részét nyújtja.

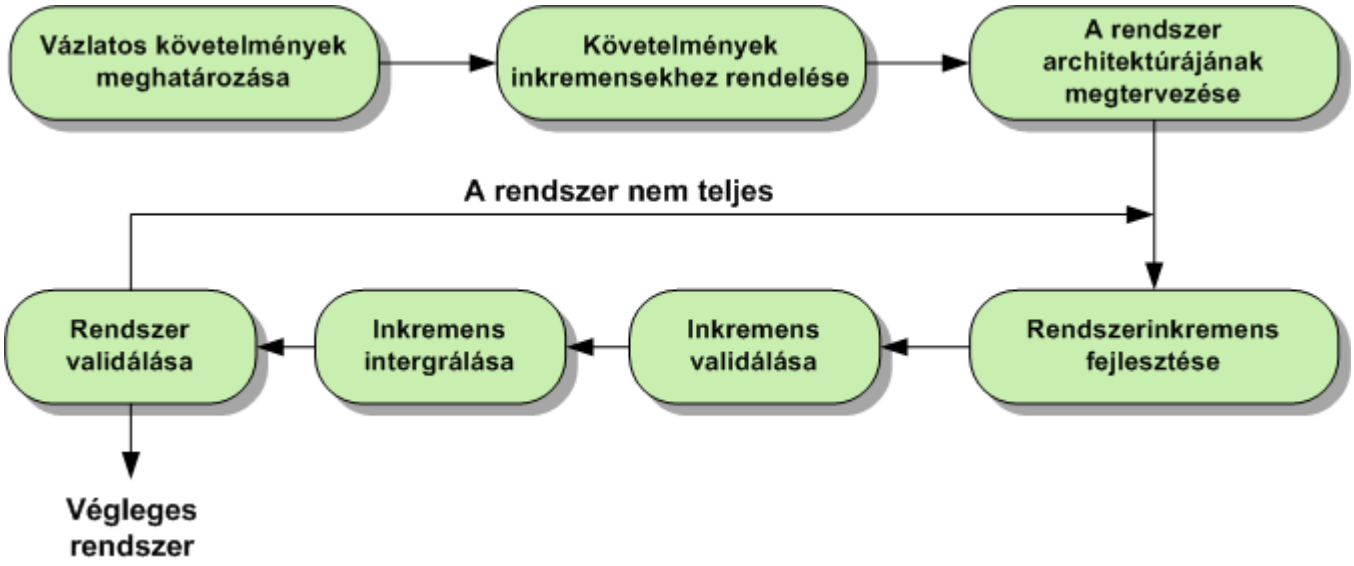
2. **Követelménymódosítás**: a fázisban a megtalált komponensek információit felhasználva elemezzük a követelményeket. Majd módosítani kell azokat az elérhető komponenseknek megfelelően. Ahol nem lehetséges a követelmény módosítása, ott újra el kell végezni a komponenselemzést és alternatív megoldást kell keresni.

3. **Rendszertervezés újrafelhasználással**: ebben a szakaszban a rendszer szerkezetének tervezését hajtjuk végre. A tervezés kulcseleme az, hogy milyen komponenseket akarunk újrafelhasználni, és úgy alakítani a szerkezetet, hogy azok működhessenek. Amennyiben nincs elérhető újrafelhasználható komponens, akkor új szoftverek is kifejleszthetők.

4. **Fejlesztés és integráció**: a nem megvásárolt, illetve átalakításra kerülő komponenseket ki kell fejleszteni és a rendszerbe integrálni. A rendszer-integráció ebben a modellben sokkal inkább tekinthető a fejlesztési folyamat részének, mint különálló tevékenységnek.

Inkrementális fejlesztés

Az inkrementális megközelítési mód egy köztes megközelítés a vízesésmodell és az evolúciós fejlesztési modellek között. A vízesésmodell megköveteli az ügyféltől, hogy véglegesítse a követelményeket mielőtt a tervezés elindulna, a tervezőtől pedig azt, hogy válasszon ki bizonyos tervezési stratégiákat az implementáció előtt. A vízesésmodell előnye, hogy egyszerűen menedzselhető, mert külön választja az egyes fázisokat. Ezzel szemben viszont olyan robosztus rendszerek jöhetnek létre, amik esetleg alkalmatlanok a változtatásokra. Az evolúciós megközelítésnél pedig megengedettek a követelményekkel és tervezésekkel kapcsolatos döntések elhagyása, ami pedig gyengén strukturált és nehezen megérthető rendszerekhez vezethetnek. Az inkerementális fejlesztés alapú megközelítést az alábbi ábrán figyelhetjük meg:



Egy inkrementális fejlesztési folyamatban a megrendelő meghatározza:

- nagy körvonalakban a rendszer által nyújtandó szolgáltatásokat,
- mely szolgáltatások fontosabban, melyek kevésbé.

A követelmények meghatározása után a követelmények inkremensekben való megfogalmazása és hozzárendelése következik. A szolgáltatások inkremensekben való elhelyezése függ a szolgáltatás prioritásától is. A magasabb prioritású szolgáltatásokat hamarabb kell biztosítani a megrendelő felé.

Inkrementális fejlesztés jellemzői

Miután az inkrementációs lépéseket meghatároztuk, az első inkrementációs lépés által előállítandó szolgáltatások követelményeit részletesen definiálni kell. Ezután pedig következik az inkremens kifejlesztése. A fejlesztés ideje alatt sor kerülhet további követelmények elemzésére, de az aktuális inkrementációs lépés követelményei nem változtathatók.

Amennyiben egy inkremens elkészült, a rendszer bizonyos funkcióit akár be is üzemeltethetik

korábban. Így tapasztalatokat szerezhetnek a rendszerrel kapcsolatban, amely a későbbi inkrementációs lépésekben segítségre lehet a követelmények tisztázásában. Amennyiben az új inkremens elkészül, azt integrálni kell a már meglévő inkremensekkel. Ezzel a rendszerfunkciók köre egyre bővül. Az inkrementációs fejlesztés előnyei:

1. *A megrendelőnek nem kell megvárnia míg a teljes rendszer elkészül*. Már az első inkremens kielégítheti a legkritikusabb követelményeket, így a szoftver már menet közben használhatóvá válik.

2. A *megrendelők használhatják a korábbi inkremenseket* mint prototípusokat, ami által tapasztalatokat szerezhetnek.

3. *Kisebb a kockázat*a annak, hogy a teljes projekt kudarcba fullad.

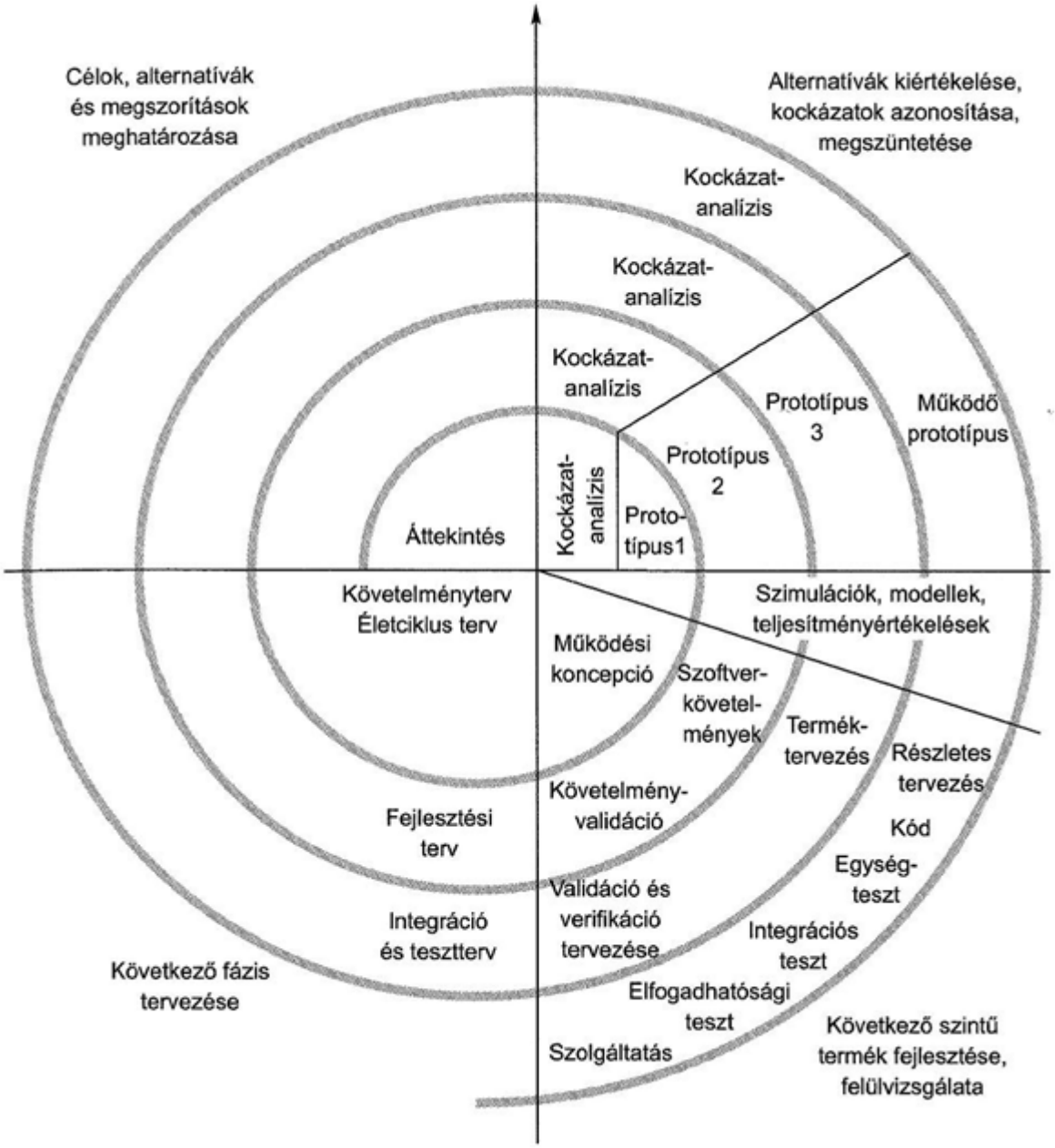
4. A fejlesztés során *a magasabb prioritású inkremenseket szállítjuk le hamarabb*, ezért mindig a legfontosabb szolgáltatások lesznek többet tesztelve. Ezért kisebb a hiba esélye a rendszer legfontosabb részeiben.

Mindezek ellenére az inkrementális fejlesztésnek is megvannak a maga hibái. Fontos, hogy az inkremensek megfelelően kis méretűk legyenek és minden inkrementációs lépésnek szolgáltatni kell valami rendszerfunkciót. Sokszor azonban nehéz a megrendelő követelményeit megfelelő méretű inkrementációs lépésekre bontani.

Spirális modell

spirális fejlesztési modellt **Boehm** javasolta először már 1986-ban a „A Spiral Model of Software Development and Enhancement” címmel megjelent publikációjában, amely azóta széles körben elterjedt az irodalomban és a gyakorlatban is.

A modell alapgondolata szintén eltér az eddigiektől, mert a szoftverfolyamatot nem tevékenységek és közöttük található esetleg visszalépések sorozataként tekinti, hanem inkább egy spirálként reprezentálja. A spirál minden egyes körben a szoftverfolyamat egy-egy fázisát reprezentálja. A spirálnak, mint a folyamatot reprezentáló vonalnak egy további sugallni valója is van. Mégpedig az, hogy tetszőleges számú kör, mint iteráció tehető meg. Azaz egy szoftver fejlesztése sohasem ér véget.



Fázisok értelmezése

A spirál minden egyes ciklusát négy szektorra oszthatjuk fel:

1. **Célok kijelölése**: az adott projektfázis által kitűzött célok meghatározása. Azonosítani kell a folyamat megszorításait, a terméket, fel kell vázolni a kapcsolódó menedzselési tervet. Fel kell ismerni a projekt kockázati tényezőit, és azoktól függően alternatív stratégiákat kell tervezni ha lehetséges.

2. **Kockázat becslése**: minden egyes felismert kockázati tényező esetén részletes elemzésre kerül sor. Lépéseket kell tenni a kockázat csökkentése érdekében.

3. **Fejlesztés és validálá**s: a kockázat kiértékelése után egy fejlesztési modellt kell választani a problémának megfelelően. Pl. evolúciós, vízesés, stb modellek.

4. **Tervezés**: a folyamat azon fázisa, amikor dönteni kell arról, hogy folytatódjon-e egy következő ciklussal, vagy sem. Ha a folytatás mellett döntünk, akkor fel kell vázolni a projekt következő fázisát.

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:infrendalapjai_architekturak:szoftvertechnologia:szoftverfolyamat_modelljei

Last update: **2024/11/13 19:32**