A **szoftverfolyamat** tevékenységek és kapcsolódó eredmények sora, amelyek egy szoftvertermék előállításához vezetnek.

Habár számos különböző szoftverfolyamat létezik, vannak olyan *alapvető tevékenységek*, amelyek minden szoftverfolyamatban közösek:

1. ***Szoftverspecifikáció***: A szoftver funkcióit, illetve annak megszorításait definiálni kell.
2. ***Szoftvertesztelés és implementáció***: A specifikációnak megfelelő szoftvert elő kell állítani.
3. ***Szoftvervalidáció***: A szoftvert validálni kell, hogy biztosítsuk, azt fejlesztettük ki, amit az ügyfél kívánt
4. ***Szoftverevolúció***: A szoftvert úgy kell alakítani, hogy megfeleljen a megrendelő kívánsága szerint történő változásoknak.

**A szoftverfolyamat modelljei**

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

1. ***Vízesés modell:*** Ez a folyamat alapvető tevékenységeit a folyamat különálló fázisaiként tekinti, ezek a fázisok a követelményspecifikáció, a szoftvertervezés, az implementáció, tesztelés..stb
2. ***Evolúciós fejlesztés:*** Ez a megközelítési mód összefésüli a specifikáció, a fejlesztés és a validáció tevékenységeit. Az absztrakt specifikációból gyorsan kifejleszthető egy kezdeti rendszer. A későbbiekben ezt a rendszert kell a megrendelő inputjából továbbfinomítani úgy, hogy az kielégítse az ügyfél kívánságait.
3. ***Komponensalapú fejlesztés:*** Ez a megközelítési mód a nagy mennyiségű újrafelhasználható komponensek létezésén alapszik. A rendszerfejlesztési folyamat ezekenek a komponenseknek rendszerré történő integrációjára összpontosít ahelyett, hogy az alapoktól fejlesszék ki azokat.

Ez a három általános modell terjedt el széles körben. Nem kizárólagos a használatuk, és gyakran keverednek is egymással, főként a nagy rendszerek fejlesztésekor.

A fenti általános modellek különféle változatait használhatják különféle szervezetek. A legfontosabb változatuk azonban talán a formális szoftverfejlesztés, ahol formális matematikai eszközökkel modellezzük a létrehozandó rendszert. pl: Cleanroom folyamat.

**A vízesés modell**

Az egyes fázisok lépcsőzetesen kapcsolódnak egymáshoz. Szakaszai:

1. ***Követelmények elemzése és meghatározása:*** A rendszer szolgáltatásai, megszorításai és céljai a rendszer felhasználóival történő konzultáció alapján alakulnak ki.
2. ***Rendszer- és szoftvertervezés:*** A rendszer tervezési folyamatában választódnak szét a hardver- és szoftverkövetelmények. Itt kell kialakítani a rendszer átfogó architechtúráját.
3. ***Implementáció és egységteszt:*** Ebben a szakaszban a szoftverterv programok, illetve programegységek halmazaként realizálódik. Az egységteszt azt ellenőrzi, hogy minden egyes egység megfelel-e a specifikációnak.
4. ***Integráció és rendszerteszt:*** Megtörténik a különálló programegységek, illetve programok integrálása és telejs rendszerként történő tesztelése, hogy a rendszer megfelel-e a követelményeknek. A tesztelés után a szoftverrendszer átadható az ügyfélnek.
5. ***Működtetés és karbantartás:*** A szoftver életciklusának leghosszabb fázisa. A karbantartásba beletartozik az olyan hibák kijavítása, amelyekre nem derült fény az életciklus korábbi szakaszaiban, a rendszeregységek implementációjának továbbfejlesztése, valamint a rendszer szolgáltatásainak továbbfejlesztése.

A fázisok eredménye egy vagy több dokumentum, amelyek jóváhagyása megtörtént. A következő fázis addig nem indulhat el, amíg az előző fázis be nem fejeződött.

Probléma: A folyamat korai szakaszaiban kell állást foglalni és elkötelezni magunkat, ezzel nehezebbé válik az ügyfél követelményeinek megváltoztatásához történő alkalmazkodás. Csak akkor használható jól, h előre jól ismerjük a követelményeket. Nagyobb rendszertervezési projekt részeinél alkalmazzák leginkább.

**Evolúciós fejlesztés**

Az evolúciós fejlesztés alapötlete az, hogy ki kell fejleszteni egy kezdeti implementációt, azt a felhasználókkal véleményeztetni, majd sok-sok verizón keresztül addig finomítani azt, míg megfelelő rendszert el nem érünk. Ez a megközelítési mód sokkal jobban érvényesíti a tevékenységek közötti párhuzamosságot.

Két különböző típusa ismert:

1. ***Feltáró fejlesztés:*** A folyamat célja az, hogy a megrendelővel együtt feltárjuk a követelményeket, és kialakítsuk a végleges rendszert. A végleges rendszer úgy alakul ki, hogy egyre több, az ügyfél által kért új tulajdonságot társítunk a már meglévőkhöz.
2. ***Eldobható prototípus készítése:*** Ekkor az evolúciós fejlesztés célja az, hogy lehető legjobban megértsük az ügyfél követelményeit, és azokra alapozva pontosabban definiáljuk azokat.

Az evolúciós hatékonyabb, mint a vízesés modell, ha olyan rendszert szeretnénk fejleszteni, amely közvetlenül megfelel az ügyfél kívánságainak. Előnye, hogy a rendszer-specifikáció inkrementálisan fejleszthető.

A vezetőség és a tervezők szepontjából két probléma merülhet fel:

1. A folyamat nem látható: A menedzsereknek rendszeresen leszállítható részeredményekre van szükségük, hogy mérhessék a fejlődést.
2. A rendszerek gyakran szegényesen struktúráltak. A folyamatos változtatások lerontják a szoftver struktúráját.

A várhatóan rövid élettartamú kis vagy közepes rendszerek esetében is (500 000 programsorig) az evolúciós megközelítési mód a legcélravezetőbb.

**Komponensalapú szoftvertesztelés**

Az elmúlt néhány évben jelent meg (2007-ben írták a könyvet) és kezdett el terjedni széles körben egy olyan szoftverfejlesztési megközelítési mód, mely az újrafelhasználáson alapszik. Az újrafelhasználás-orientált megközelítési mód nagymértékben az elérhető újrafelhasználható szoftverkomponensekre, illetve azok egységes szerkezetbe történő integrációjára támaszkodik.

Szakaszai:

1. ***Komponenselemzés:*** Adott a követelményspecifikáció, és keresést kell végrehajtani, hogy mely komponensek implementálták azokat.
2. ***Követelménymódosítás:*** Elemezni kell a követelményeket, a megtalált komponenesek információit felhasználva. Ezek után módosítani kell azokat az elérhető komponenseknek megfelelően. Ahol a módosítás lehetetlen, ott újra a komponenselemzési tevékenységet kell elővenni, és alternatív megoldást keresni.
3. ***Rendszertervezési újrafelhasználással:*** A rendszer szerkezetét kell megtervezni vagy egy már meglévő vázat felhasználni. Ha nincsenek elérhető komponensek, akkor új szoftverek is kifejleszthetők.
4. ***Fejlesztés és integráció:*** A nem megvásárolható komponenseket ki kell fejleszteni és a COTS rendszerekkel egy rendszerbe integrálni.

Előnye, hogy csökkenti a kifejlesztendő szoftverek számát, így csökkentve a költségeket, illetve a kockázati tényezőket. A rendszer gyorsabban leszállítható. Viszont a követelményeknél elkerülhetetlenek a kompromisszumok, amelyek oda vezethetnek, hogy a rendszer nem felel meg a felhasználó valódi kívánságainak.

A CBSE kezd elterjedni a webszolgáltatásokat nyújtó rendszerek fejlesztése esetén.

**Projektmenedzsment**

A szoftverprojekt-menedzsment fontos része a szoftvertervezésnek. A jó menedzsment persze nem garantálja a projekt sikerét, de a rossz menedzsment általában a projekt sikertelenségét okozza. A szoftvert későn szállítják le, a költségek meghaladják az eredetileg tervezett költségeket, és az is előfordulhat, hogy nem fognak megfelelni a követelményeknek.

A **szoftvermenedzser** felelőssége megtervezni és ütemezni a fejlesztési projektet. Biztosítják a szabványok szerinti végrehajtást, valamint megfigyelik és folyamatosan ellenőrzik, hogy a fejlesztés megfelelő időterv és költségvetés szerint halad.

Vezetői tevékenységek:

* indítványokat írni
* a projektet megtervezni és ütemezni
* a projekt költségét figyelemmel kísérni
* a projektet figyelni és felülvizsgálni
* a személyzetet kiválasztani és fejleszteni
* beszámolókat írni és előadni

**A projekt tervezése**

A terv típusai:

* ***Minőségi terv:*** meghatározza azokat a minőségi eljárásokat és szabályokat, amelyeket a projektben használni kell.
* ***Validációs terv:*** Meghatározza a megközelítési módot, az erőforrásokat és az ütemterveket, amelyeket a rendszer validációjához használni kell.
* ***Konfigurációkezelési terv:*** Meghatározza a konfigurációkezelési eljárásokat és szerkezeteket, amelyeket alkalmazni kell.
* ***Karbantartási terv:*** Előre megadja a rendszer karbantartási követelményeit, a karbantartás költségeit és a szükséges erőfeszítéseket.
* ***Munkaerő-fejlesztési terv:*** Meghatározza, hogyan kell a projektcsapat tagjainak szaktudását és tapasztalatait fejleszteni.

**Projekt tervezése**

Megállapítani a projekt megszorításait.

A projekt paramétereinek egy kezdeti összegzését elkészíteni.

Definiálni a projekt mérföldköveit és részeredményeit.

**while** a projekt még nincs kész vagy még nem vonták vissza **loop**

Felvázolni a projekt ütemtervét.

Elindítani a tevékenységeket az ütemterv alapján.

Várakozni (egy darabig)

Átvizsgálni a projekt előrehaladását

Felülvizsgálni a projekt paramétereinek becslését

Frissíteni a projekt ütemtervét

Újratárgyalni a projekt megszorításait és részeredményeit

**if** (probléma merül fel) **then**

Elindítani a technikai felülvizsgálatokat és a lehetséges átdolgozásokat

**end if**

**end loop**

**A projektterv**

A projektterv felsorolja a projekthez rendelkezésre álló erőforrásokat, a munka felosztását és a munka végrehajtásának ütemtervét is. Néhány szervezetben a projektterv egy egyszerű dokumentum, amely minden egyes különböző típusú tervet magában hordoz. Más esetekben a projektterv kizárólag a fejlesztési folyamatokkal foglalkozik. Tartalmaz még egyéb tervekre történő hivatkozásokat, de a tervek maguk külön találhatóak meg.

Minden egyes tervnek ajánlott tartalmaznia a következő szakaszokat:

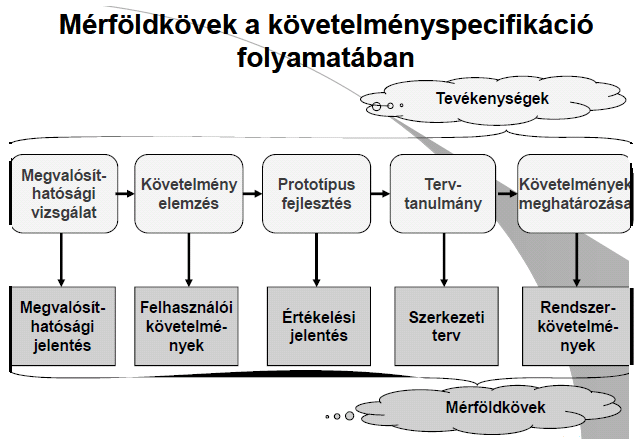
1. ***Bevezetés:*** Ez a projekt céljainak tömör leírása, felsorolja a megszorításokat is (pl költségvetés, idő stb.), amelyek befolyásolják a projekt menedzselését.
2. ***Projektszervezet***: Megadja a projektcsapat összeállításának módját, a részt vevő embereket és azok szerepét.
3. ***Kockázatelemzés:*** Leírja a projekt lehetséges kockázatait, azok bekövetkezésének valószínűségét és a kockázat csökkentésének ajánlott stratégiáját.
4. ***Hardver- és szoftverforrás-követelmények:*** Azt kell megadni, hogy milyen hardver és milyen támogatott szoftver szükséges a fejlesztéshez. Ha a hardvert meg kell vásárolni, megbecsüli annak árát és a beszerzésének határidejét.
5. ***Munka felosztása:*** Megadja a projekt tevékenységekre történő felosztását, és azonosítja a tevékenységekhez kapcsolódó mérföldköveket és részeredményeket.
6. ***Projekt ütemterve:*** Meghatározza a tevékenységek közötti függőségeket, megbecsüli a mérföldkövek eléréséhez szükséges időket, és javaslatot tesz a tevékenységekhez rendelkező emberekre.
7. ***Figyelési és jelentéskészítési mechanizmusok:*** Meghatározza a menedzser által előállítandó jelentéseket, azok előállításának idejét és a használandó projektfigyelési, monitorozási technikát.

**Mérföldkövek és részeredmények**

Mivel a szoftver kézzel nem fogható, ezért az információk csak dokumentum formájában biztosíthatók, amelyek a fejlesztés alatt álló szoftver állapotát írják el. Ezek nélkül az információk nélkül lehetetlenné válik megítélni a haladást, megbecsülni a költségeket és frissíteni az ütemtervet.

Sok mérföldkövet kell meghatározni, ahol minden egyes mérföldkő egy szoftverfolyamat-tevékenység végpontja. A részeredmények már átadhatóak a megrendelőnek.

A részeredmények, leszállítható részterméket általában mérföldkövek, de a mérföldkövek nem szükségszerűen részeredmények. A mérföldkövek lehetnek belső projekteredmények, melyeket a projektmenedzser arra használ, hogy ellenőrizze a projekt előrehaladását, de azokat nem adják át a megrendelőnek.



**Projekt ütemezése:**

A menedzserek megbecsülik a tevékenységek elvégzéséhez szükséges időt és előforrásokat, és egy összefüggő szekvenciába rendezik őket.

A projekt ütemezése magába fogalja a projekt teljes munkájának különálló tevékenységekre bontását és a tevékenységek elvégzéséhez szükséges idő megítélését is. Általában ezek közül a tevékenységek közül néhány párhuzamosan is elvégezhető. A projekt ütemezőinek kell koordinálniuk ezeket a párhuzamos tevékenységeket, és úgy megszervezniük a munkát, hogy a munkaerő kihasználtsága optimális legyen. El kell kerülni az olyan szituációkat, ahol a teljes projekt azért csúszik, mert egy kritikus feladat még nincs befejezve.

Az ütemtervek megállapításakor a menedzsereknek nem ajánlatos azt feltételezniük, hogy a projekt minden egyes szakasza problémamentes lesz. Az egyének megbetegedhetnek, eltávozhatnak, a hardverek meghibásodhatnak, vagy az elengedhetetlen szoftvereket és hardvereket későn szállíthatják. Ha a projekt még új és technikailag is újszerű, bizonyos részeiről kiderülhetnek, hogy nehezebbek és hosszabb ideig tartanak, mint azt gondoltuk.

A menedzsereknek mindig meg kell becsülniük minden egyes tevékenységhez a szükséges erőforrásokat és naptári időt. Akkor jó a saccolás, ha semmiben sem tévedünk, és a becslésünk kiterjed a nem várt problémákra is.

A projekt ütemtervét legtöbbször diagramok halmazaként reprezentálják, amelyek a munka felosztását, a tevékenységek függőségeit, a munkaerők elhelyezkedését stb. szemléltetik.

**Oszlopdiagramok és tevékenységhálók**

Grafikus jelölések, melyeket a projekt ütemtervének reprezentálására használunk.

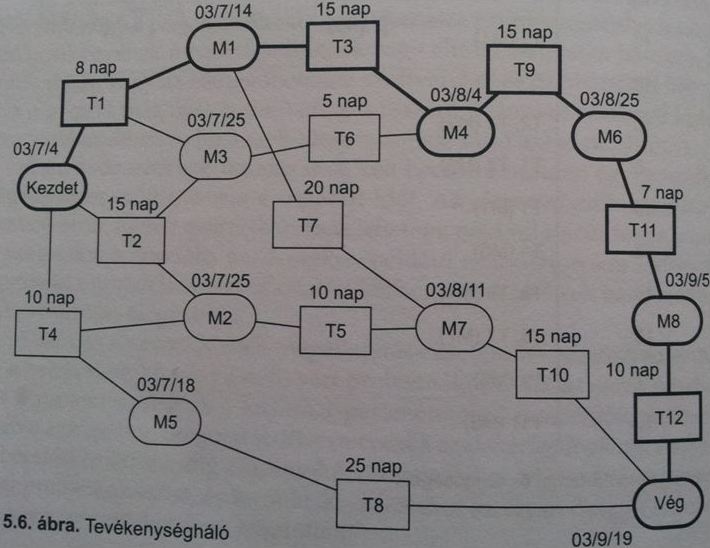
Az **oszlopdiagramok** azt mutatják meg, hogy ki a felelős a tevékenységért, és a tevékenységet mikorra ütemezték be, azaz mikor kezdődik és mikor ér véget.

A **tevékenységhálók** a projektet felépítő különböző tevékenységek közötti függőségeket ábrázolják.

*Tevékenységháló*

A tevékenységekhez adott függőségek és becsült időtartamuk ismeretében már könnyen generálható a tevékenységháló, amely a tevékenységek sorrendjét mutatja. Az is látszódik, hogy mely tevékenységeket lehet párhuzamosan végrehajtani, és mit kell egymás után.

A tevékenységeket téglalapokkal, a mérföldköveket és a projekt részeredményeit pedig lekerekített sarkú téglalapokkal jelöljük. A hálót balról jobbra és fentről lefelé kell olvasni.



Egy tevékenység akkor kezdődhet el, mikor az azt megelőző mérföldkövet elértük. Mielőtt áthaladnánk egyik mérföldkőről a másikhoz, az összes odavezető utat be kell járnunk.

A projekt befejezéséhez szükséges legrövidebb idő megbecsléséhez tekintsük a tevékenységháló leghosszabb útját (kritikus út). Itt 11 hét, vagy 55 munkanap. A projekt teljes ütemterve függ a kritikus úttól. Bármelyik kritikus tevékenység teljesítésében bekövetkező időbeli csúszás az egész projekt késéséhez vezet.

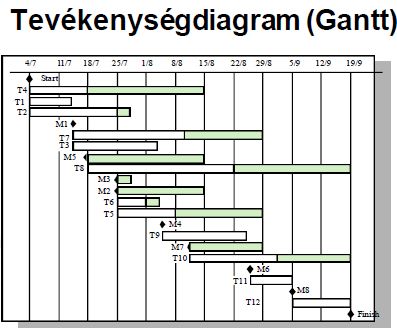
Azok a tevékenységek, amelyek nem a kritikus úton vannak, nem okoznak csúszást, amennyiben a késések hosszával kiegészítve a tevékenységeket nem haladják meg a kritikus utat.

*Tevékenységdiagram*

Az ütemezési információk reprezentációjának egyetlen lehetséges módja.A projekt naptárát ábrázolja a tevékenységek kezdeti és befejezési idejével.

Az ábrán látható sötétített oszlopok a teljesítési időbeli rugalmasságot mutatják. Ha egy tevékenységet nem sikerül időben befejezni, az a kritikus útra nincs hatással addig, amíg a tényleges befejezés ideje a sötétített oszloppal jelölt időn belülre esik.

Azoknak a tevékenységeknek, amelyek a kritikus útra esnek, nem lehet hibájuk, azaz nincs hibahatáruk (nem követi őket sötétített oszlop).



**Kockázatkezelés**

A kockázatok fenyegetik a fejlesztett projektet, a szoftvert és a szervezetet is.

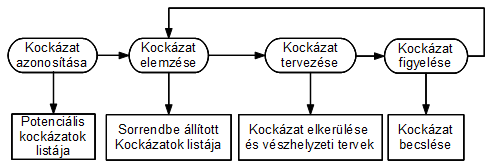
Kockázati kategóriák:

1. ***Projektkockázat***: Olyan kockázat, amely kihatással lehet a projekt ütemtervére vagy az erőforrásokra. Erre példa lehet egy tapasztalt tervező hiánya. *(pl: munkaerő elpártolása, vezetőség megváltoztatása, hardver elérhetetlensége, követelmények megváltoztatása, specifikáció késessé, méret alábecslése)*
2. ***Termékkockázat***: Olyan kockázat, amely a fejlesztett szoftver minőségére vagy teljesítményére gyakorol hatást. Ilyen például, ha egy beszerzett komponens nem nyújtja az elvárt teljesítményt. *(pl: követelmények megváltoztatása, specifikáció késessé, méret alábecslése, CASE-eszköz alulteljesítése*)
3. ***Üzleti kockázat:*** Olyan kockázat, amely a szoftver beszerzését vagy fejlesztését végző szervezetre van hatással. Például ha a konkurens egy új terméket vezet be. *(pl: technológia megváltoztatása, termékverseny)*

Természetesen ezek nem egymást kizáró kockázatok. Ha egy programozó elhagyja a csapatot, az 1-es és 2-esbe is beleillik.

**A kockázatkezelés folyamata:**

1. ***Kockázat azonosítása:*** a lehetséges projekt-, termék- és üzleti kockázatokat azonosítani kell
2. ***Kockázat elemzése:*** A kockázatok valószínűségét és következményeit fel kell becsülni
3. ***Kockázat tervezése:*** Terveket kell készíteni a kockázatokhoz azok elkerülésére és a projektre kifejtett hatásuk csökkentése érdekében.
4. ***Kockázat figyelése:*** A kockázatot folyamatosan fel kell becsülni és a kockázatok enyhítésére adott terveket folyamatosan át kell vizsgálni, ahogy egyre több információ válik elérhetővé a kockázatokról.



**Szoftvermérés**

Termék vagy folyamat valamely jellemzőjét numerikusan kifejezni (metrika).

Két csoportra lehet osztani.

* Vezérlési metrikák: Folyamatokkal kapcsolatosak, pl egy hiba javításához szükséges átlagos idő. (**Folyamat és projekt metrikák**)
* Prediktor metrikák: Termékkel kapcsolatosak, pl: LOC, osztály metódusainak száma. (**Termék metrikák**)

Mindkettő befolyásolja a vezetői döntéshozatalt.

**Termékmetrika:**

* Dinamikus: szorosabb kapcsolat egyes minőségi jellemzőkkel (teljesítmény, hibák száma)
* Statikus:
  + Közvetett kapcsolat
  + Számtalan metrikát ajánlottak már

Többféle fajta is lehet. Méretben, komplexitásban, OOP-vel kapcsolatos metrikák stb.

**Folyamat és projekt metrika:**

* Folyamat mutatók:
  + Az aktuális folyamat hatékonysága
  + Menedzsment látja mi megy jól és mi nem
  + Személyes és publikus mértékek
* Projekt mutatók:
  + Jelenlegi projekt státusza
  + Projekt team képességeinek értékelése
    - Szoftverminőség
    - termelékenység
  + Méret alapú (LOC), funkció pont alapú.

Minőségi és hatékonysági adatokat gyűjthetünk össze. Összetudjuk vetni régebbi adatokkal és tudjuk figyelni a trendeket.

Általában a régi projektek mérési adatait használjuk fel a következő **költség- és időbecslésére.**

Méret alapú metrikák esetén haszálhatunk LOC, KLOC, Hibák/emberhónap, LOC/emberhónap… sokféle mérést.

Mérhetünk különböző **Minőségi metrikákat is** pl.

* Helyesség (hiányosság/KLOC)
* Karbantarthatóság (Mean time to change)
* Integritás: A külső támadások elleni védelem.
* DRE(defect removal efficiency) = DRE = E/(E+D)
  + E a hibák száma amit az átadás előtt fedeztünk fel
  + D pedig az átadás utáni felhasználók által riportolt.