



MISKOLCI EGYETEM
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR

Szoftvertesztelés
Szoftverfejlesztési modellek

GEIAL31H-B

Tompa Tamás
egyetemi tanársegéd
Általános Informatikai Intézeti Tanszék

Miskolc, 2019

A szoftverkrízis

- A tesztelés szükségességét a **szoftverkrízis** húzta alá
- Szoftverkrízis (NATO-Szoftverfejlesztés Konferencia, 1968)
 - a szoftver projektek jelentős része sikertelen
 - Sikertelen mert:
 - vagy a tervezettnél drágábban készül el (over budget),
 - vagy a tervezettnél hosszabb idő alatt (over time),
 - vagy nem az igényeknek megfelelő,
 - vagy rossz minőségű / rossz hatásfokú / nehezen karbantartható,
 - vagy anyagi / környezeti / egészségügyi kárhoz vezet,
 - vagy átadásra sem kerül

-> a tesztelés a minőségi problémákra ad választ, a károkozás megelőzésében segít

Tesztelés a szoftverélelciklus folyamatában

- **Szoftver élelciklusa** (Software Development Life Cycle, SDLC)
 - A szoftver átadása után újabb igények merülnek fel
 - szoftver továbbfejlesztése
 - ciklikusan megújul
 - ez az élelciklus
 - az élelciklus fázisait a **módszertanok** határozzák meg
 - új igény felmerülése
 - első és utolsó lépés -> ciklikusság
 - egy hasznos szoftver élelciklusa
 - elvileg végtelen
 - gyakorlatilag előregszik (programozási nyelv, technológia, környezet stb.)

Életciklus



Módszertanok

- **Feladata**

- szoftver **élelciklus lépéseinek és azok sorrendjének meghatározása**
- egyfajta szabálykönyv
 - milyen dokumentumokat kell előállítani
 - milyen szoftvert és hogyan stb.
- a **projekt céljaitól és feladataitól függ** az adott módszertan kiválasztása

- Vízesés modell

- V-modell

- Prototípus modell

- Inkrementális fejlesztés

- Gyors alkalmazásfejlesztés

- Agilis fejlesztés

- Extrém programozás

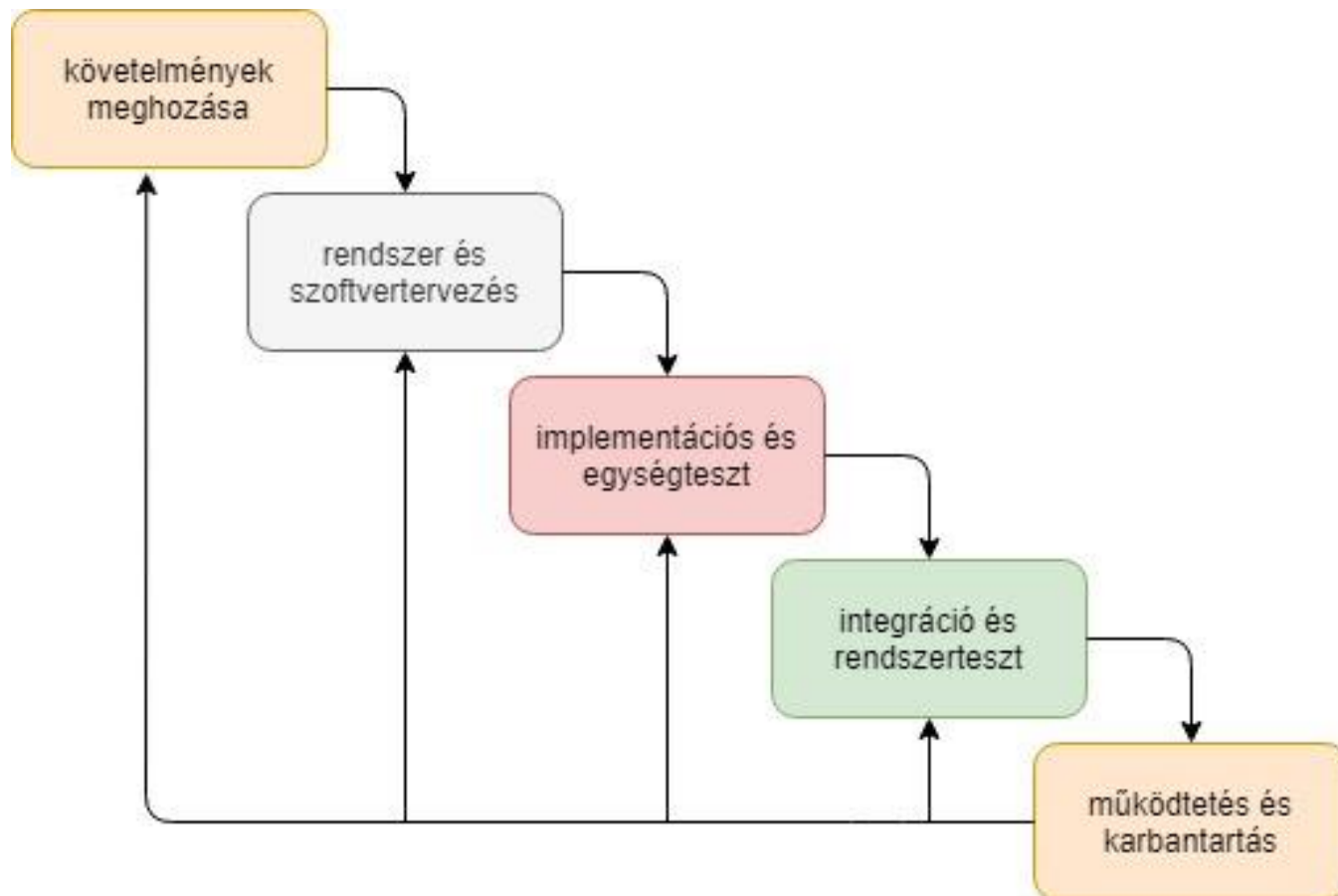
Vízesés modell

- A szoftverfejlesztés folyamatának első publikált modellje, más tervezői modellekből származik
- egyszerű
- fázisok, eredménye egy dokumentum
- egy fázis csak akkor indulhat, ha az előző befejeződött
- rugalmatlan, korai szakaszokban komoly döntések
- 1. fázis: követelmények elemzése és meghozása
 - a rendszer felhasználóival való konzultáció alapján kialakul a:
 - rendszer szolgáltatásai,
 - megszorításai,
 - célja
- 2. fázis: rendszer - és szoftverter
 - a rendszer átfogó architektúrájának kialakítása

Vízesés modell

- 3. fázis: implementáció és egységteszt
 - ebben a szakaszban megvalósul a szoftverterv
- 4. fázis: integráció és rendszerteszt
 - a különálló programegységek, programok integrálása
 - teljes rendszerként való tesztelése
- 5. fázis: működtetés és karbantartás
 - a később kiderült hibák javítása
 - a rendszeregységek implementációjának továbbfejlesztése
 - új követelmények léphetnek fel, így szükséges lehet a rendszer szolgáltatásainak továbbfejlesztése.

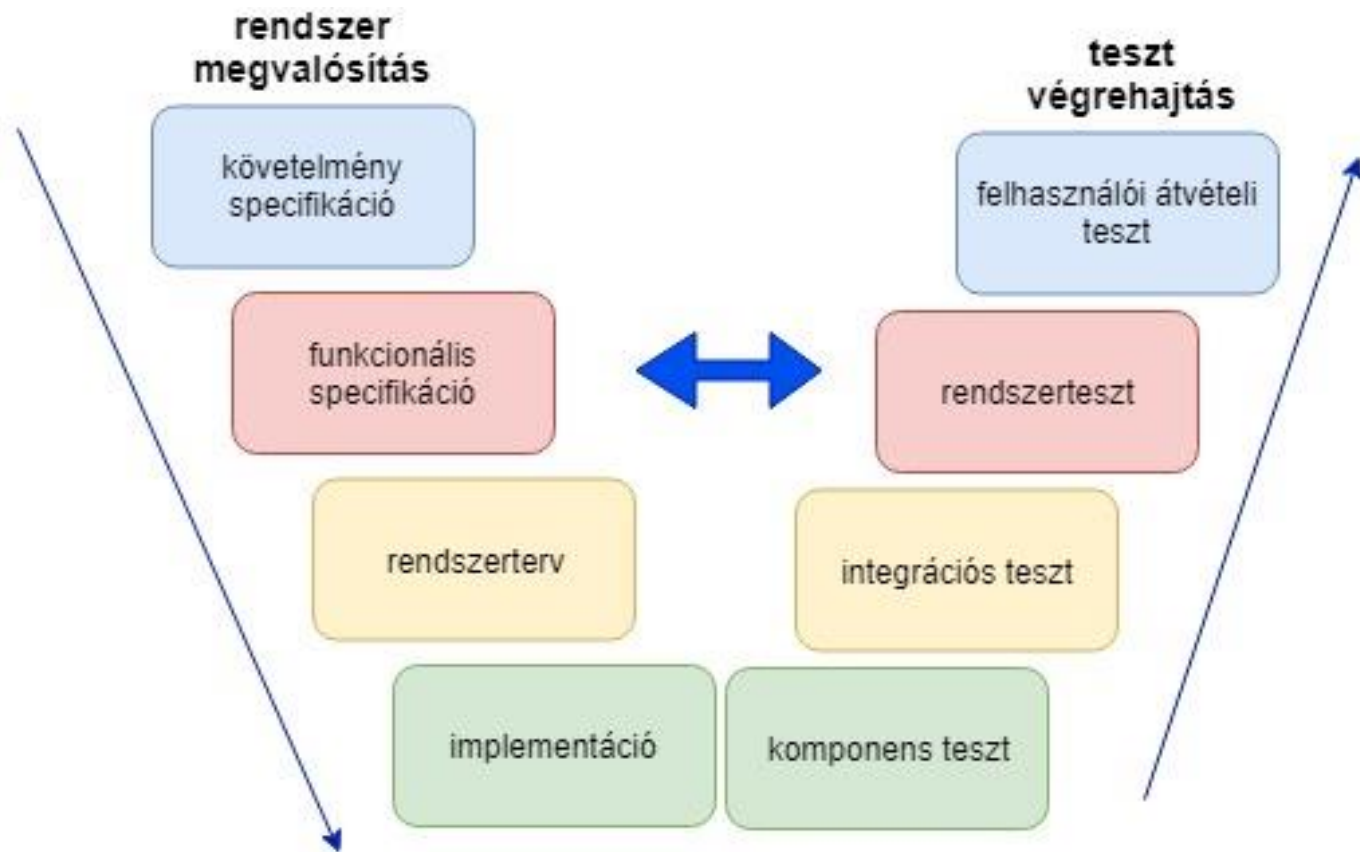
Vízesés modell



V-modell

- Két szár -> V betűhöz hasonló alak
 - fejlesztési rész -> vízesés modell
 - tesztelési rész
 - a vízesés modell kiegészítése teszteléssel
 - fejlesztés majd tesztelés
 - ha a tesztelés hibát ad akkor vissza a fejlesztéshez
 - az egy szinten lévő fejlesztési és tesztelési lépések összetartoznak
 - a tesztelési lépés a fejlesztési lépés során létrejött dokumentumokat használja, vagy a létrejött terméket teszteli
- elterjedt de nagyon merev
- ha nem változnak a követelmények a fejlesztés alatt akkor jó módszertan
- ha változnak akkor más módszer inkább -> iteratív vagy agilis módszertan

V-modell



V-modell

○ Kritikák

- a vízesés modellből ered
- rugalmatlan, képtelen jól alkalmazkodni a változásokhoz
- helytelenül, lineáris képet fest a szoftver fejlesztés folyamatáról
- a tesztelést is rugalmatlan folyamatként kezeli, ahelyett, hogy a tesztelőkre bízná a legjobb módszer felkutatását az adott probléma vizsgálatához
- sok változatának köszönhetően sok a bizonytalanság

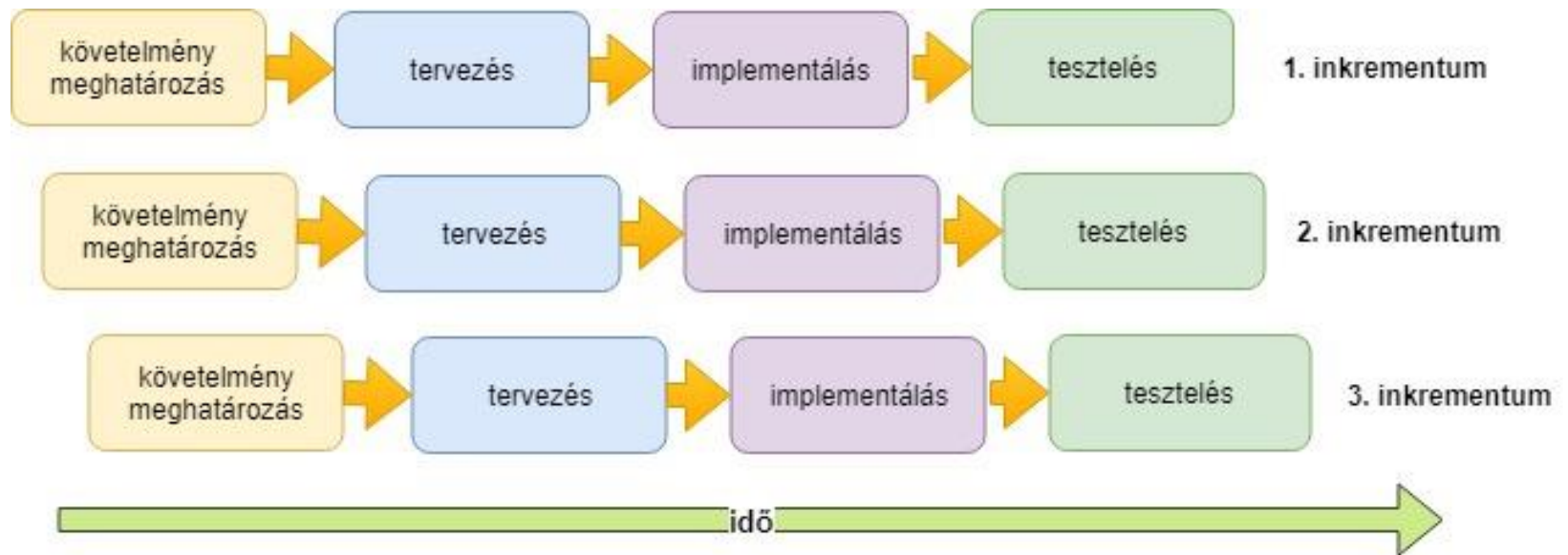
V-modell - Teszt

- teszt végrehajtása az adott szint megvalósítása után
- de nem jelenti azt, hogy a tesztelők munkája a fejlesztések lezárultával kezdődik
- amint előállnak a funkcionális és nem-funkcionális követelmények, kezdődhet az átvételi tesztesetek és a megfelelőségi elvárások tervezése
- programegységek esetében pedig az interfészeik specifikációja alapján egységtesztek készítése

Iteratív-inkrementális módszertanok

- a fejlesztés kisebb iterációk sorozata
- minden iterációban tervezés és implementáció
- életciklus lépései nem egymás után jönnek, mint a strukturált módszertanok esetén, hanem időben átfedik egymást
- folyamatos finomítás
 - minden iteráció kiegészíti a már kifejlesztett prototípust
- a folyamatra teszik a hangsúlyt azaz az iterációra
- kiegészítés hozzáadásával növekvő részrendszer jön létre, amelyet tesztelni kell
- mai módszertanok nagy része ebbe a családba tartoznak
 - prototípus modell
 - gyors alkalmazásfejlesztés (RAD)
 - Rational Unified Process (RUP)
 - agilis fejlesztési modellek

Iteratív-inkrementális módszertanok



Iteratív-inkrementális módszertanok - Teszt

- V-Modellhez hasonlóan
 - az idő elteltével egyre mélyebb szinten a teszttervezési és konkrét tesztelési feladatok
- ellenőrzés, visszajelzés és javaslattétel megjelenése
- tesztelők bevonása be a fejlesztési folyamatba amint lehet
- építés az ellenőrző munkájukra a fejlesztés minden szintjén
- bizonyított megoldásokat hozhatnak a projekt korai fázisaiban
 - tapasztalat alapján
- de: folyamatos változások -> rosszul strukturált rendszer

Prototípus modell

○ Vízesés modell

- a rendszerrel a felhasználó csak a projekt végén találkozik
- gyakran csak ekkor derült ki, hogy félreértették egymást
- probléma orvosolása - > **prototípus modell**

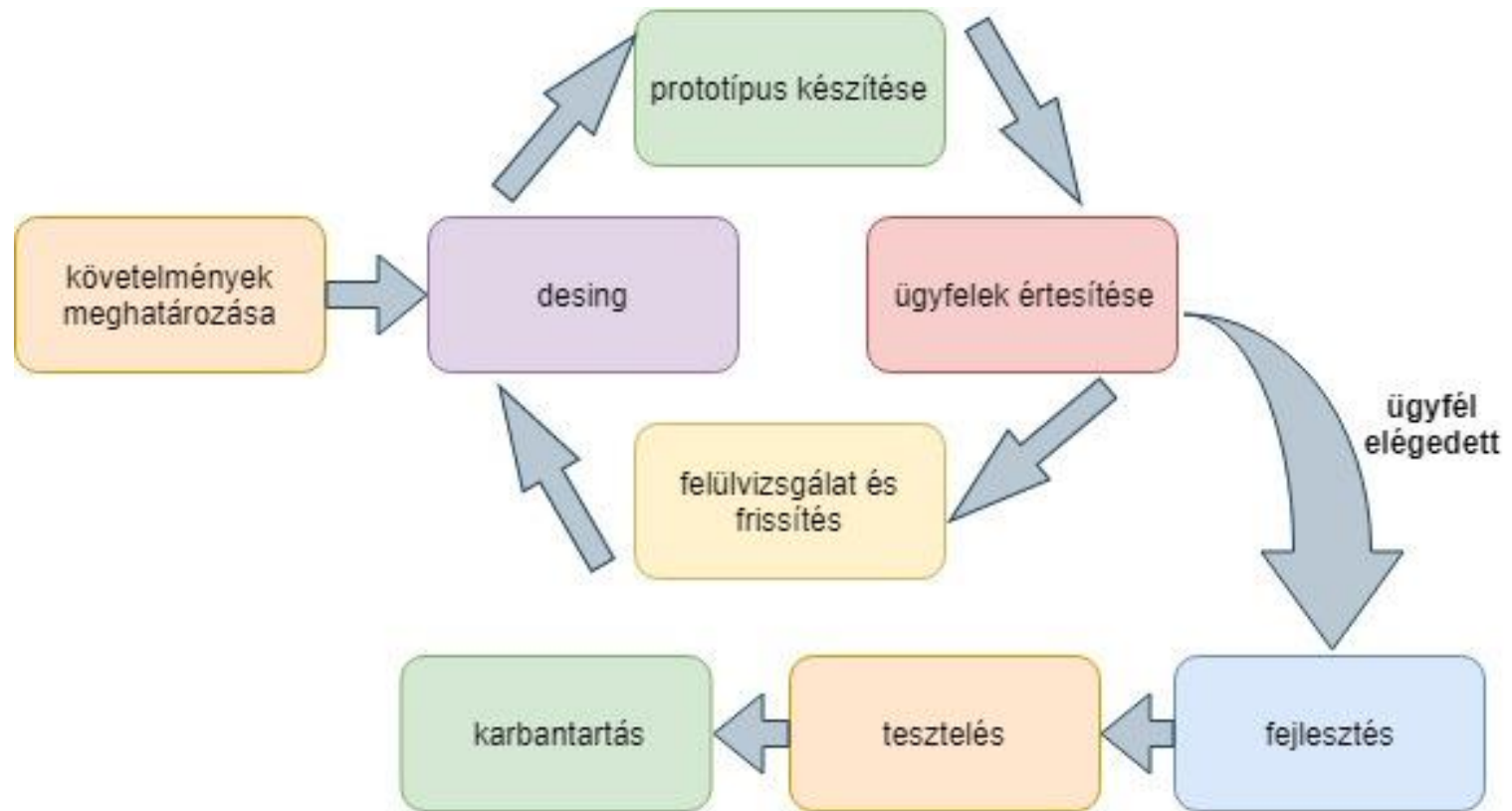
○ Prototípus modell

- a megrendelő üzleti folyamatai, követelményei nem ismerhetők meg teljesen
- ezek idővel változnak
- követelmények finomítása prototípusok segítségével
- prototípus használta után annak megfogalmazása, hogy miért nem felel meg a követelményeknek
- ezt követően finomítás
- a rendszer és a felhasználó között sok lesz a párbeszéd

Prototípus modell

- **1. lépés:** alapkövetelmények meghatározása
- **2. lépés:** kezdeti prototípus kifejlesztése: Csak a felhasználói felületeket, a mögötte lévő funkciók nem
- **3. lépés:** Bemutatás: Ez egyfajta felhasználói átvételi teszt.
- **4. lépés.** A követelmények pontosítása: A visszajelzéseket felhasználva pontosítjuk a követelmény specifikációt. Ha még mindig nem elég pontos a specifikáció, akkor a prototípust továbbfejlesztjük és ugrunk a 3. lépésre.

Prototípus modell



Agilis szoftverfejlesztés

- iteratív szoftverfejlesztési módszer
- 2001 (Agile Manifesto kiadvány)
- szó jelentése: sürgetés
 - olyan adottság, amely képes reagálni a változásokra
- résztvevők alkalmazkodása a projekthez
- Szemlélet szerint:
 - értékesebbek az **egyének** és interaktivitás szemben a **folyamatokkal** és az eszközökkel
 - értékesebb a **működő szoftver** szemben a terjedelmes **dokumentációval**
 - értékesebb az **együttműködés** a megrendelővel szemben a szerződéses **tárgyalásokkal**
 - értékesebb az **alkalmazkodás** a változásokhoz szemben a **terv követésével**

Agilis szoftverfejlesztés

○ Alapelvek

- legfontosabb a megrendelő kielégítése
- a követelmények kései változtatása sem okoz problémát
- működő szoftver / prototípus átadása rendszeresen, a lehető legrövidebb időn belül
- napi együttműködés a megrendelő és a fejlesztők között
- a leghatékonyabb kommunikáció a szemtől-szembeni megbeszélés
- az előrehaladás alapja a működő szoftver
- folyamatos figyelem a technikai kitűnőségnek
- egyszerűség, a minél nagyobb hatékonyságért
- önszervező csapatok készítik a legjobb terveket

Agilis szoftverfejlesztés

- Pl.

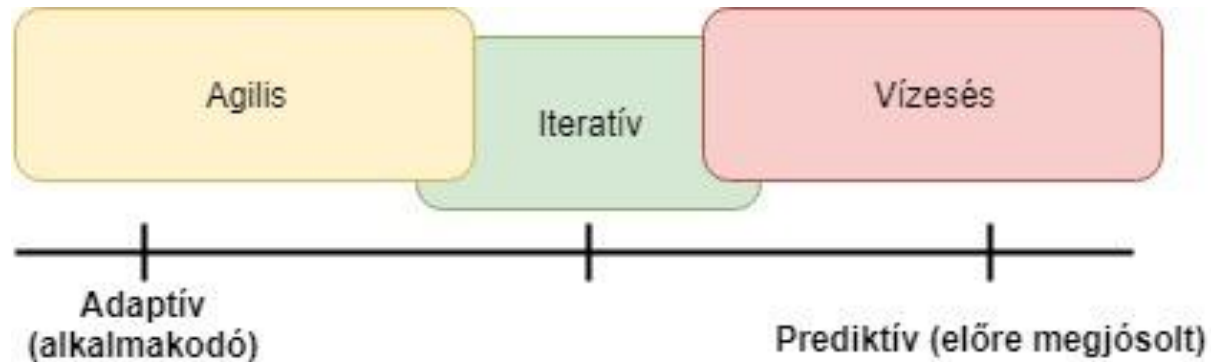
- **Scrum**

- Extrém Programozás (XP)

- Közös jellemzők

- kevesebb dokumentáció
 - növekvő rugalmasság, csökkenő kockázat
 - könnyebb kommunikáció, javuló együttműködés
 - a megrendelő bevonása a fejlesztésbe
 - hónapok helyett hetek
 - fontosak a határidők
 - nem flexibilisek

Módszertanok - összehasonlítás



- Átfedés a módszerek között
- Agilis <--> vízesés
- Adaptív
 - hosszútávú tervezés, nincs előre jóslás
 - követlen problémákra koncentrálni
 - „mit fogunk a héten csinálni”
- Prediktív
 - megtervezett lépések
 - minden lépés az egészre összpontosítva

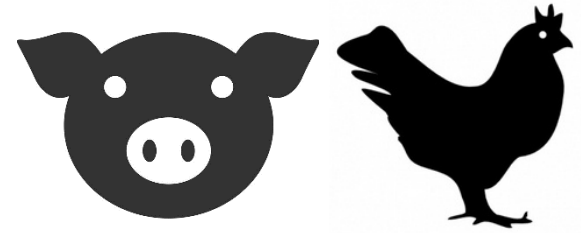
Scrum

- Rögbiből átvett kifejezés
 - jelentése: visakodik, összecsap
 - 1990
- megmaradt sport szemlélet
 - Pl. váltófutás
 - stafétát a szoftver
 - a fázisok a futók
 - ha egy futó rosszul teljesít akkor az egész csapat veszít
- Alapgondolat
 - fázisok erősen átlapoltak
 - kisebb csoportok, más területek emberiből
 - az összes fázisban együtt dolgoznak

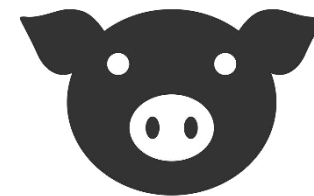
Scrum - jellemzők

- Adaptív tulajdonságok
- Nincs „forgatókönyv”
- A fejlesztő csapat egyszerre kezd el dolgozni
 - a csapat felelős a végeredményért
 - Scrum Team
- Kommunikáció
 - különböző szakterületek
- Átlátható, világos
 - ki, miért és milyen határidővel felelős

Scrum - szerepkörök



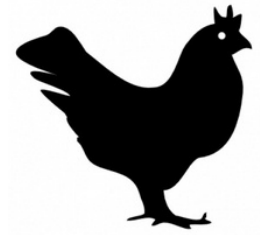
- A disznó és a csirke mennek az utcán
- a csirke megszólal:
 - „Te, nyissunk egy éttermet!”
- disznó:
 - „Jó ötlet, mi legyen a neve?”
- a csirke erre gondolkozik, majd azt feleli:
 - „Nevezzük Sonkás-tojásnak!” (Ham and eggs)
- a disznó erre:
 - „Nem tetszik valahogy, mert én biztosan mindent beleadnék, te meg éppen csak hogy részt vennél benne.”



Scrum – „Disznók”

- „vérüket” adják a projekt sikeréért
- elkötelezettek a szoftver projekt sikerében

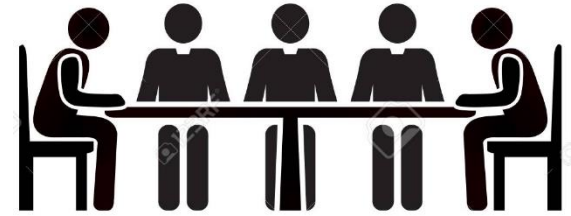
- Disznók
 - Scrum mester (Scrum Master)
 - működtetés, folyamat felügyelése
 - Terméktulajdonos (Product Owner)
 - megrendelő, felelős azért, hogy a csapat mindig azt a részét fejlessze a terméknek, amely éppen a legfontosabb
 - Csapat (Team)
 - 5-9 fős csapat, különböző területekről
 - feladatok megvalósítása
 - csapatjáték fontos -> „passzolgatás”



Scrum – „Csirkék”

- közvetetten részei a folyamatnak
- Csirkék
 - Üzleti szereplők (Stakeholders)
 - megrendelők, forgalmazók
 - Menedzsment (Managers)
 - környezet felállítása a csapatok számára
 - megfelelő, legjobb környezet

Scrum – megbeszélések

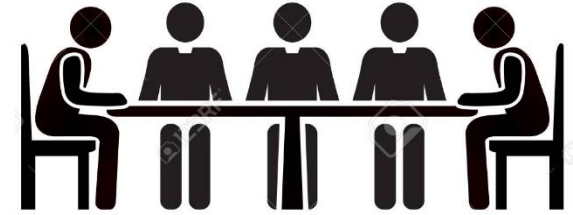


- Sprint Planning Meeting (futamtervező megbeszélés)
 - ki mennyi munkát tud elvállalni, majd ennek tudatában dönti el a csapat, hogy mely sztorikat vállalja be a következő sprintre

- Backlog Grooming/Backlog Refinement
 - a Product Backlog finomítása a Teammel együtt, előfordulhat például, hogy egy taszk túl nagy, így story lesz belőle, és utána taszkokra bontva lesz feldolgozva

- Daily Meeting/Daily Scrum
 - a sprint ideje alatt minden nap kell tartani egy rövid megbeszélést, ami maximum 15 perc
 - Mit csináltál a tegnapi megbeszélés óta?
 - Mit fogsz csinálni a következő megbeszélésig?
 - Milyen akadályokba ütköztél az adott feladat megoldása során?

Scrum – megbeszélések



- Sprint Review Meeting (Futam áttekintés)
 - minden sprint végén összeülnek a szereplők, és megnézik, hogy melyek azok a sztorik, amelyeket sikerült elkészíteni, illetve az megfelel-e a követelményeknek

- Sprint Retrospective (Visszatekintés)
 - az egyik legfontosabb meeting. A Scrum egyik legfontosabb funkciója, hogy felszínre hozza azokat a problémákat, amelyek hátráltatják a fejlesztőket a feladatmegoldásban -> alkalmazkodás a feladatokhoz



Scrum – dokumentumok

- Story
 - lényegi leírás a megrendelőktől
- Product backlog (termék teendő lista)
 - Story feldolgozása, priotásokkal
- Sprint backlog (futam teendő lista)
 - a konkrét feladatok feltüntetése az adott sprintre (ki, mit, milyen határidővel vállalt be)
- Burn down chart (Napi Eredmény Kimutatás)
 - diagram, amely segít megmutatni, hogy az ideális munkatempóhoz képest hogyan halad a csapat az aktuális sprinten belül

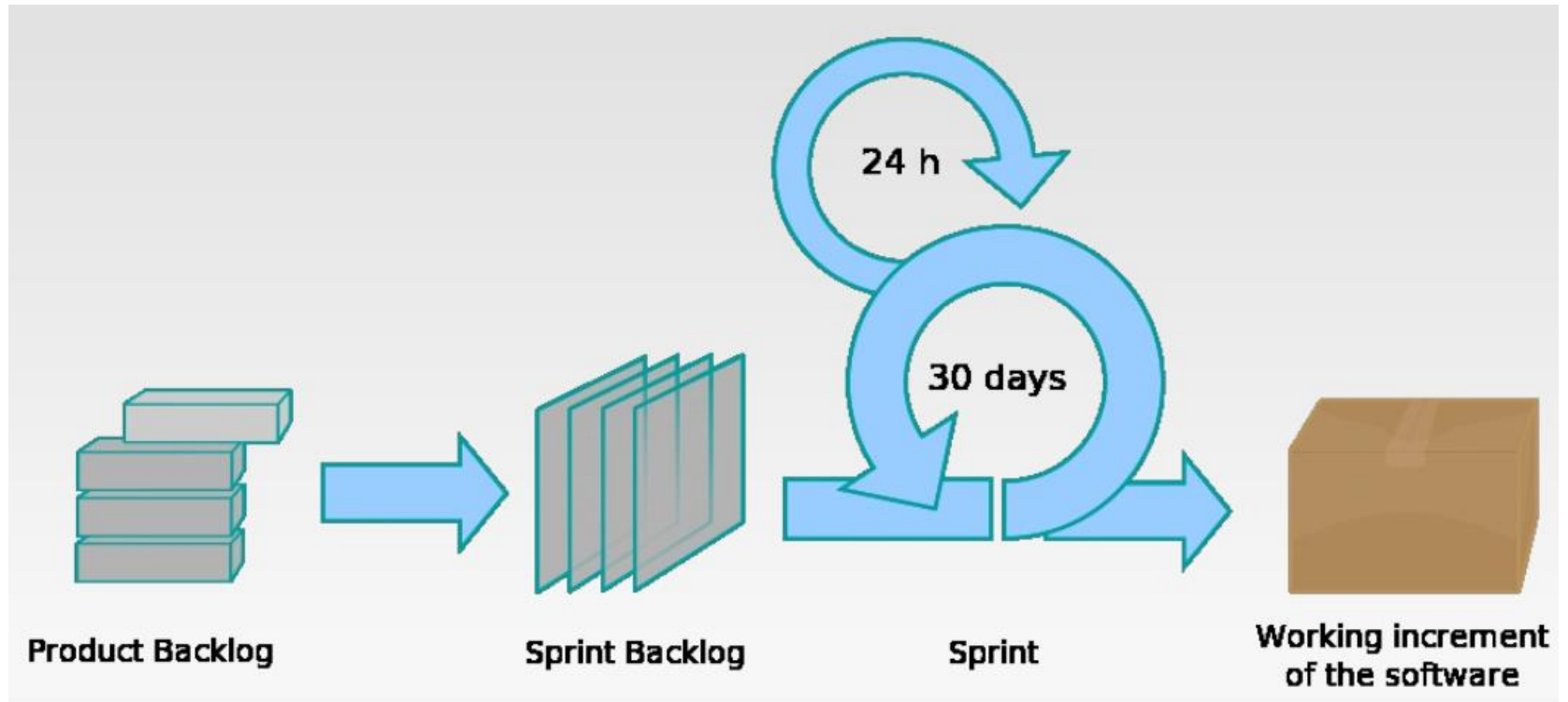


Scrum – fogalmak

- Sprint (futam)
 - előre megbeszélte hosszúságú fejlesztési időszak, általában 2-4 hétig tart
 - iterációs ciklus -> addig kell ismétetni amíg el nem tűnnek a megoldásra váró felhasználói sztorik
 - minden sprint végére leszállítható szoftver előállítása

- Akadály (Impediment)
 - olyan gátló tényező, amely a munkát hátráltatja
 - csak és kizárólag munkahelyi probléma tekinthető akadálynak (a csapattagok magánéleti problémái nem azok)
 - akadály például, hogy lejárt az egyik szoftver licence
 - a Scrum Masternek kell elhárítani az akadályokat

Scrum



Extrém programozás



- Extreme Programming, XP
- agilis módszertan
- az eddigi módszertanokból átveszi a jól bevált technikákat
 - és azokat nem csak jól, hanem extrém jól alkalmazza
 - majd minden mást feleslegesnek tekint
- != „programozzunk összeesésig” módszerrel
 - 24 órás vagy akár 48 órás programozó verseny

Extrém programozás



○ 4 tevékenység

● Kódolás

- legfontosabb, előjönnek a nehézségek, kommunikációs a fejlesztők között -> mindenki ugyanazt érti alatta

● Tesztelés

- addig nem lehetünk benne biztosak, hogy egy funkció működik, amíg nem teszteltük. Az extrém felfogás szerint kevés tesztelés kevés hibát talál, extrém sok tesztelés megtalálja mind

Extrém programozás



○ 4 tevékenység

● Odafigyelés

- a fejlesztőknek oda kell figyelniük a megrendelőkre, meg kell érteniük az igényeiket

● Tervezés

- tervezés nélkül nem lehet szoftvert fejleszteni, mert az ad-hoc megoldások átláthatatlan struktúrához vezetnek. Mivel fel kell készülni az igények változására, ezért úgy kell megtervezni a szoftvert, hogy egyes komponensei amennyire csak lehet függetlenek legyenek a többitől.

Extrém programozás



○ Jellemző technikák

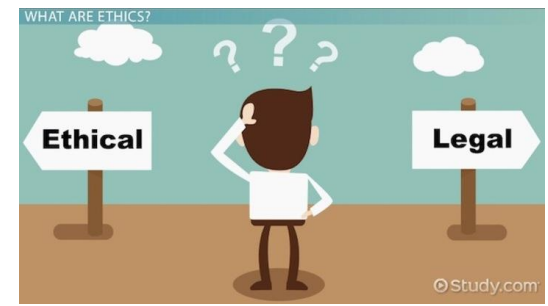
- Páros programozás (pair programming)
 - ketten egy kód, ír<-> figyel, ha hibát lát szól
- Teszt vezérelt fejlesztés (test driven development)
 - először egységteszt írás majd metódus implementálás
- Forráskód átnézés (code review)
 - vezető fejlesztő átnézi a kódot -> hogyan kell jobban csinálni
- Folyamatos integráció (continuous integration)
 - verziókövető rendszerbe bekerült kódok integrációs tesztje
- Kódszépítés (refactoring)
 - a már letesztelt, működő kódot lehet szépíteni, de funkcionalitás nem változhat

Etikai kódex



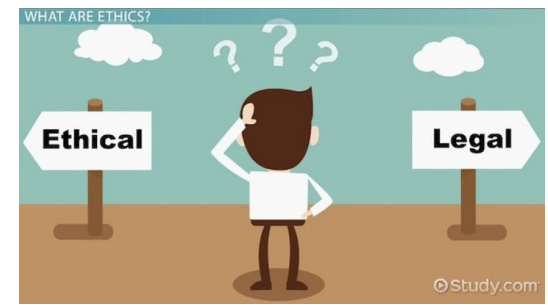
- A szoftverek tesztje során a résztvevők bizalmas információkhoz juthatnak hozzá
 - az etikai kódexre azért van szükség, hogy az információkat ne használják fel illetéktelenül
- A kódex pontjai a következők:
 - **KÖZÉRDEK** – A képesített tesztelőknek következetesen a közérdeknek megfelelően kell tevékenykedniük.

Etikai kódex



- **MEGRENDELŐ ÉS MUNKAADÓ** – A képesített szoftvertesztelőknek úgy kell tevékenykedniük, hogy a megrendelőik, illetve munkaadóik igényeit legjobban kiszolgálják, ugyanakkor a közérdekkel ne kerüljenek szembe.
- **TERMÉK** – A képesített szoftvertesztelőknek biztosítaniuk kell, hogy az általuk tesztelt, átadásra kerülő termék, vagy rendszer megfelel a legmagasabb szakmai szabványoknak.
- **VÉLEMÉNY** – A képesített szoftvertesztelőknek feddhetetlennek és függetlennek kell maradniuk a szakmai véleményalkotáskor.

Etikai kódex



- **MENEDZSMENT** – A képesített szoftver tesztmenedzsereknek és vezetőknek azonosulniuk és támogatniuk kell az etikai kódexet a szoftvertesztelés menedzsmentje felé.
- **SZAKMA** – A képesített szoftvertesztelőknek elő kell segíteniük a szakma hírnevét és feddhetetlenségét a közérdeknek megfelelően.
- **MUNKATÁRSAK** – A képesített szoftvertesztelőknek korrektül és támogatóan kell fellépni a munkatársaikkal szemben és segíteniük kell a szoftverfejlesztőkkel való együttműködést.
- **SZEMÉLYES** – A képesített szoftvertesztelőknek életük végéig tanulniuk kell a szakmájukban és munkájuk végzése során figyelniük kell arra, hogy az etikai kódex a munkájuk részévé váljon.



Gyakorlat

Gyakorlat

- Írjon egy „List” nevű osztályt, amely emberek (people, String, ArrayList) tárolására alkalmas
- Legyen „add” metódusam amely által egy embert lehet hozzáadni
- Legyen „remove” metódus, amely által egy embert lehet törölni
 - ha nincs benne a törlendő ember -> NoSuchElementException
- Legyen „size” metódusa, amely visszaadja az emberek számát
- Legyen „isEmpty” metódusa, amely visszaadja, hogy üres-e a tároló vagy sem
- Legyen „removeAll” metódusa, amely kiűríti a „people” nevű tárolót

Gyakorlat

- Írjon egy „ListTest” nevű osztály, amely az előzőleg implementált „List” nevű osztály metódusait teszteli
- Minden teszt lefutása előtt legyen feltöltve a tároló emberekkel (init)
 - @Before
- Írjon a lista méretének ellenőrzésére vonatkozó tesztesetet (size)
- Írjon a lista ürességének ellenőrzésére vonatkozó tesztesetet (isEmpty)
 - assertTrue/assertFalse
- Írjon új elem hozzáadására vonatkozó tesztesetet
- Írjon a lista tartalmát törölő metódusra vonatkozó tesztesetet (removeAll)
 - törlés majd méret ellenőrzés

Gyakorlat

- Írjon olyan tesztesetet, amely nem létező ember töltésére vonatkozik
 - exception!
- Írjon olyan metódust, amely minden teszt után lefut és kiüríti a listát (destroy)
 - @After



Köszönöm a figyelmet!

thank you 😊