



# Szoftvertchnológia

2.Népszerű életciklus modellek.  
Hagyományos és agilis  
szoftverfejlesztés.

BSc Course

Dr. Katalin Balla



# Tartalom

- Szoftverfejlesztési / technikai / műszaki folyamatok
- Népszerű élelciklus modellek
  - vízesés, V-modell, spirál, iteratív, incrementális
- Hagyományos és agilis megközelítések
  - Agilis és Lean alapelvek
- Konfliktusban van-e egymással a hagyományos és az agilis szoftverfejlesztés?



# Szoftverfejlesztési folyamatok

- Software Engineering Processes – magyarul nevezzük őket még technikai / műszaki / mérnöki folyamatoknak is. A továbbiakban **a szoftverfejlesztés műszaki folyamataiként** említjük őket.
- Általában minden szakember egyetért abban, hogy a szoftverfejlesztés az alábbi folyamatokat tartalmazza:
  - ☐ Követelmények fejlesztése és menedzsmentje
  - ☐ Tervezés
  - ☐ Kódolás
  - ☐ Tesztelés
  - ☐ Átadás
  - ☐ Karbantartás
- Ezzel együtt, nem létezik egyetlen, mindenki által, mindig és mindenütt elfogadott megközelítése a szoftvermérnöki / szoftverfejlesztési tevékenységeknek (lásd 1. előadás).
- A szoftverfejlesztési tevékenységeket általában szoftverfejlesztési életciklus modellekben írják le. (Software Development Life Cycle models (SDLC)).



# Szoftver életciklus

- Egy időperiódus, ami akkor kezdődik, amikor a szoftverterméket kigondoljuk és akkor fejeződik be, amikor a szoftvert már nem használják. A szoftver életciklus jellemzően a következő fázisokat tartalmazza: koncepció, követelmény, tervezés, megvalósítás, teszt, installáció és ellenőrzés, operáció és üzemeltetés, valamint időnként egy leállítási fázist.
  - Megjegyzés: a fázisok átfedhetnek egymást, vagy akár ismétlődhetnek is
  - Forrás: Szoftvertesztelés egységesített kifejezéseinek gyűjteménye
  - [http://www.hstqib.com/images/d/d8/HTB-Glossary-3\\_2.pdf](http://www.hstqib.com/images/d/d8/HTB-Glossary-3_2.pdf)




# Szoftver életciklus modellek

- A szoftverfejlesztésben alkalmazott életciklus modelleket a következő kategóriákba szokás sorolni:
  - Szekvenciális modellek
  - Inkrementális modellek
  - Iteratív modellek



# Szoftver életciklus modellek

- A szoftverfejlesztési modelleket a konkrét projekt és termék jellegzetességeinek függvényében testre kell szabni
- Mindig a projekt céljának, a készülő termék típusának, a termék működésével kapcsolatos kockázatoknak megfelelő életciklus modellt kell kiválasztani a szoftverfejlesztéshez.
  - Például, egy kis, belső adminisztráció segítő szoftvert másképpen kell elkészíteni, mint egy nagyobb, biztonságkritikus rendszert (pl. egy autó fékrendszerét vezérlő szoftvert). Az alkalmazott életciklus modell is különböző lesz.
  - ISTQB Foundation Level Syllabus alapján



# Szoftvermérnöki / szoftverfejlesztési módszertanok

- Megadják annak a módját, ahogyan a problémákat meg lehet oldani; az elvégzendő feladatok sorozatát is leírják.
- Irányt mutatnak , támogatás nyújtanak a munka során.
- Felhasználják a korábbról összegyűlt tapasztalatot hasonló problémák megoldására.
- Nagyobb valószínűséggel elérjük célunkat, ha követjük ajánlásaikat.
- Elnevezés:
  - ☐ Életciklus modell vagy módszertan ?



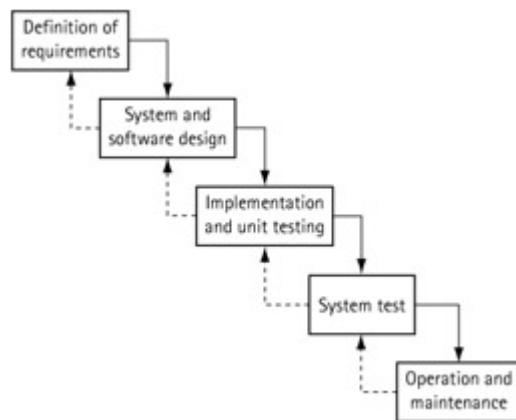
# Szekvenciális életciklus modellek

- A szoftverfejlesztés folyamatát lineáris, szekvenciális folyamatként ábrázolják.
  - Ez azt jelenti, hogy minden fázis csak akkor kezdődhet, ha az előző fázis befejeződött; nem lapolódnak át a fázisok.
  - Két ismert példa: vízesés modell és V-modell.



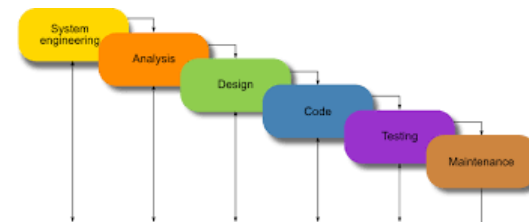
# Szekvenciális modellek

## Boehm klasszikus vízesés modellje



[Boehm76] **B. W. Boehm:** Software engineering.

*IEEE Transactions on Computers* C-25 (12) 1226–1241, 1976.



<https://airbrake.io/blog/sdlc/waterfall-model>



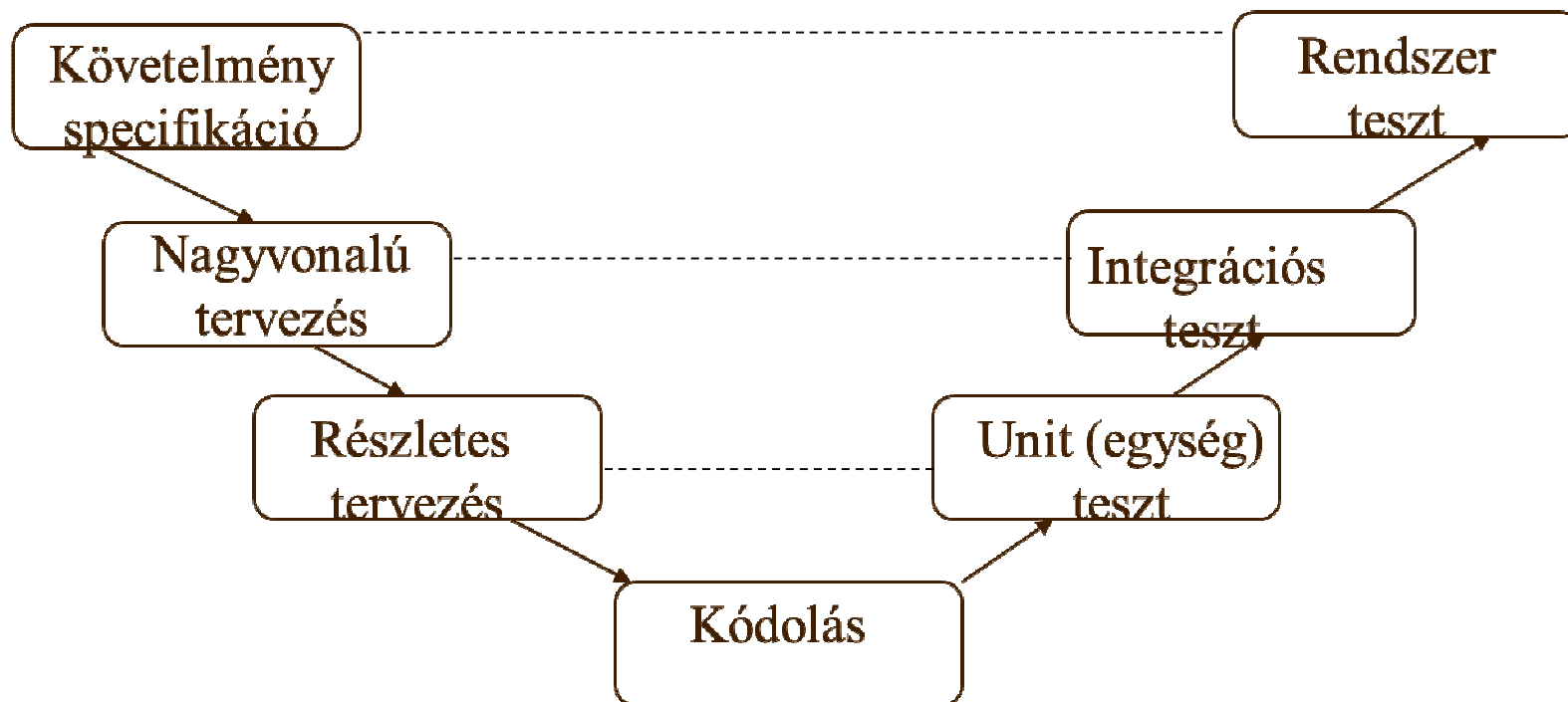
[https://www.youtube.com/watch?v=\\_ZKvvaZEFKE](https://www.youtube.com/watch?v=_ZKvvaZEFKE)



<https://medium.com/synapse-india/waterfall-model-of-software-development-a-sure-fire-practice-for-your-professional-software-needs-4c8997419800>

# Szekvenciális modellek

## ■ A V-modell



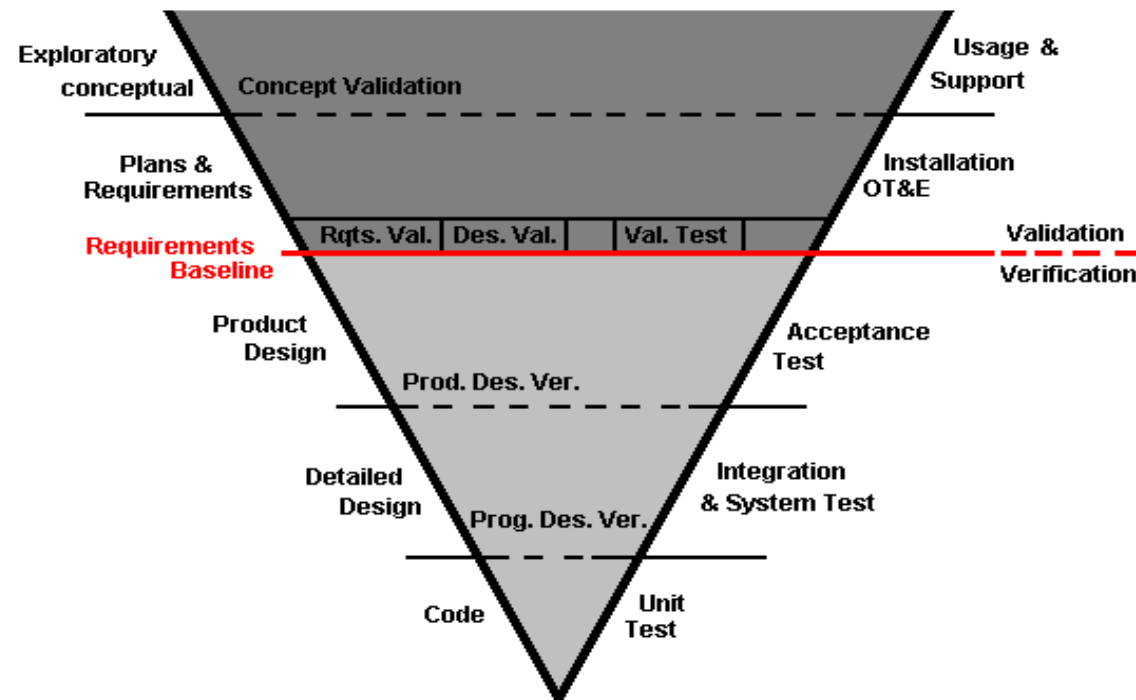
Egy adott fázisban megjelenített információkat felhasználjuk az adott fázishoz tartozó teszt esetek létrehozásában. Tesztelésben: V-modell



## Még több információ a V-modellről

- [/Myers, 1979/](#) The Art of Software Testing
  - The testing cycle has been structured to model the development cycle.
- [/Boehm, 1979/](#) Guidelines for Verifying and Validating Software Requirements and Design Specifications
  - "V-chart" which shows the context of verification and validation activities throughout the software lifecycle
- [/VM 1997/](#) V-Model 97, Lifecycle Process Model
  - Lifecycle Process Model -Developing Standard for IT Systems of the Federal Republic of Germany. General Directive No. 250. June 9
  - V-model: Regulations setting up all activities, products, and their logical interdependencies during the development and maintenance/modification of systems, realizing the system tasks predominantly by using IT, within the scope of the federal administration.
- [/Sommerville, 1999/](#) Software Engineering
  - V&V Process: is a whole life-cycle process. V&V must be applied at each stage in the software process.

# Még több információ a V-modellről



[Boehm, 1979/](#) Guidelines for Verifying and Validating Software Requirements and Design Specifications



- (Sommerville, 1999/ )



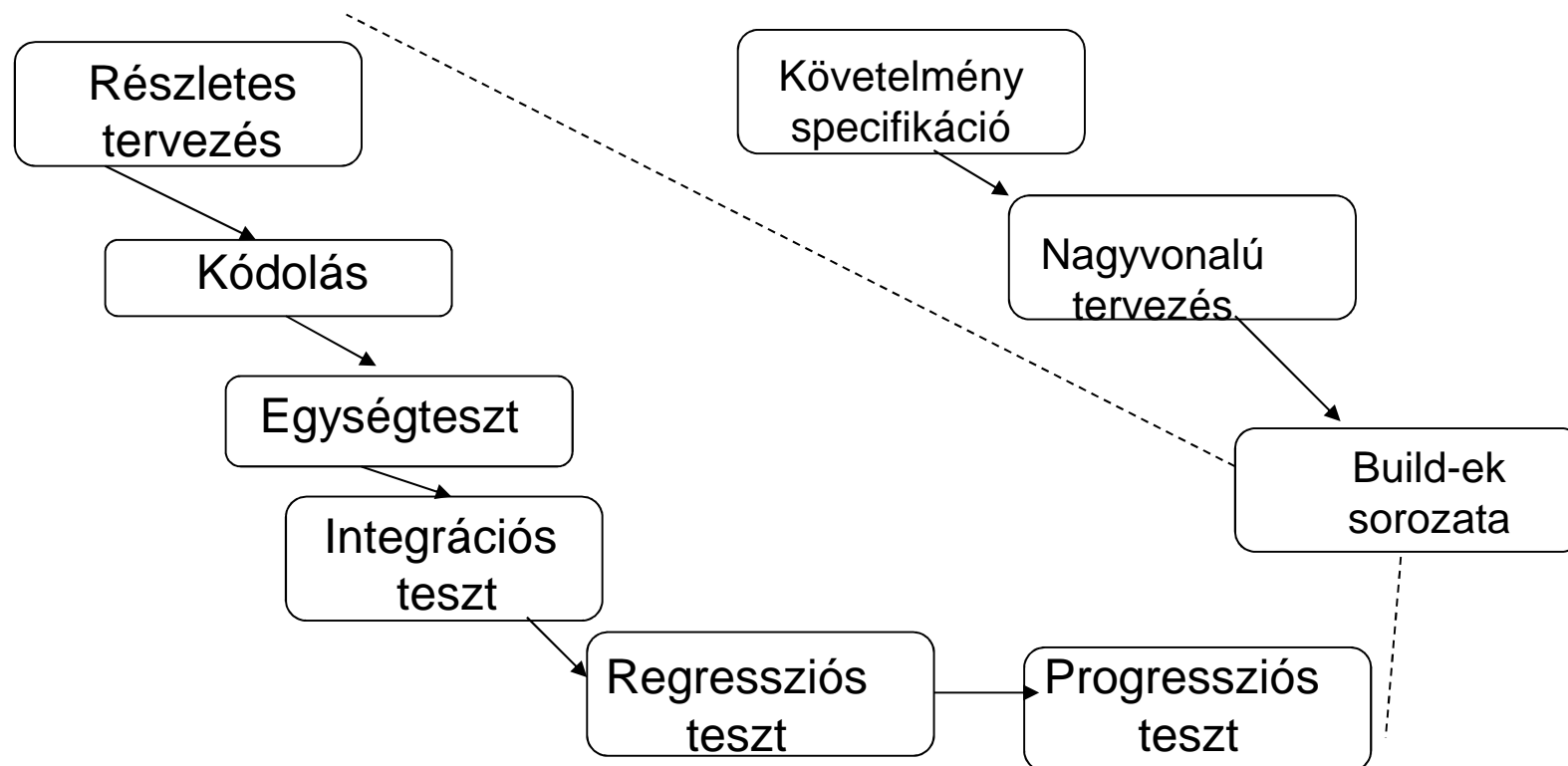


# Inkrementális modellek

- A követelményeket kisebb csoportokra bontják, és a követelményelemzés tervezés ,kódolás, tesztelés erre a kisebb követelménycsoportra történi, rövidebb idő alatt. A rövidebb fejlesztési ciklusokat inkrementumoknak nevezzük.
- Példák inkrementális modellekre: RUP, RAD, Spiral.
  - A termék “build”-jei inkrementálisan fejlődnek.
  - Mindig más a “build” meghatározása.
  - A felhasználó hamarabb kap visszajelzést.

# Inkrementális modellek

## Példák





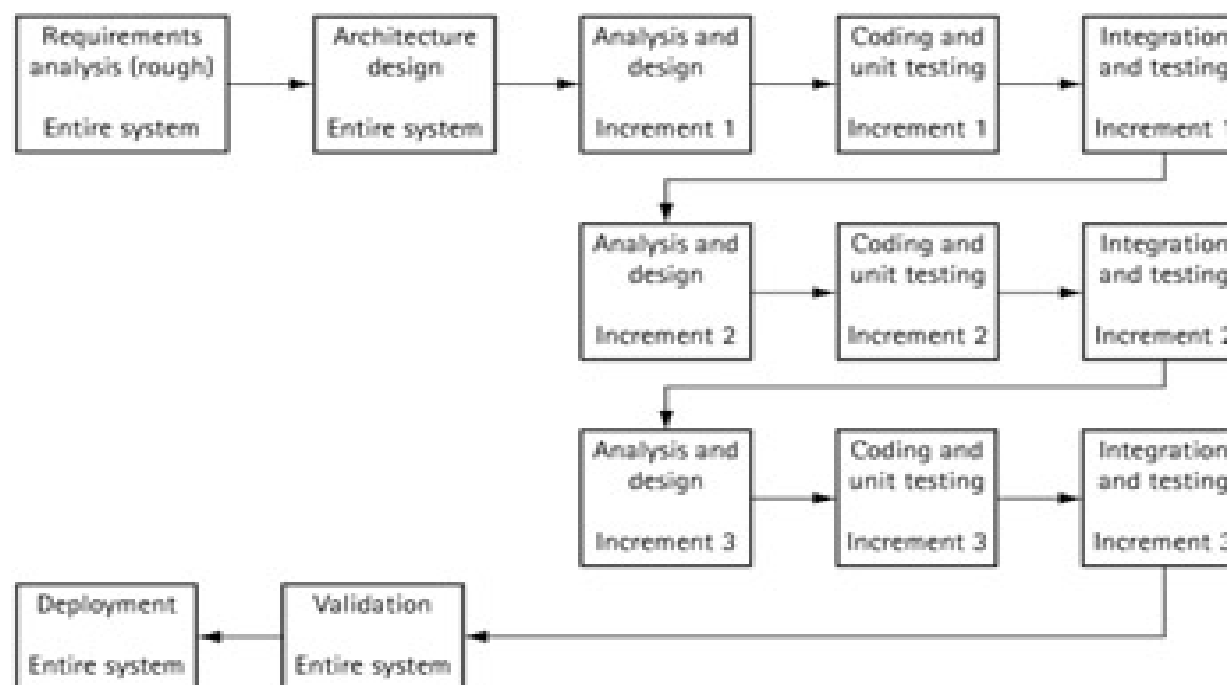
# Inkrementális fejlesztésnél...

- *Regressziós teszteléssel* bizonyítjuk, hogy ami az előző build-ben működött, a jelenlegi build-ben is működik
- *Progressziós tesztelés*: feltételezzük, hogy az integrációs teszt rendben lefutott, és az új funkciókat tesztelhetjük



# Inkrementális modellek

## Példák



[Jacobson99] **Ivar Jacobson, Grady Booch, and James Rumbaugh:**  
*The Unified Software Development Process*, Object Technology Series.

Boston: Addison-Wesley, 1999



The diagram illustrates the software development lifecycle, organized into three main horizontal sections: **Definiálás** (Definition), **Fejlesztés és verifikáció** (Development and Verification), and **Értékelés** (Evaluation).

- Definiálás (Definition):** Includes **Célok** (Goals), **Alternatívák** (Alternatives), and **Korlátok** (Constraints). It leads to **Specifikáció tervezése** (Specification Design).
- Fejlesztés és verifikáció (Development and Verification):** This section contains the core development phases:
  - rendszer tervezés** (System Design)
  - Implementáció tervezése** (Implementation Design)
  - Működés tervezése** (Operation Design)
  - Definiálás** (Definition)
  - Tervezés** (Design)
  - Implementálás** (Implementation)
- Értékelés (Evaluation):** Includes **Kockázat elemzés** (Risk Analysis) and **Prototípusok készítése** (Prototyping). The prototyping phase is further divided into **működő** (Working) and **1**, **2** (iterations).

**Flow and Feedback Loops:**

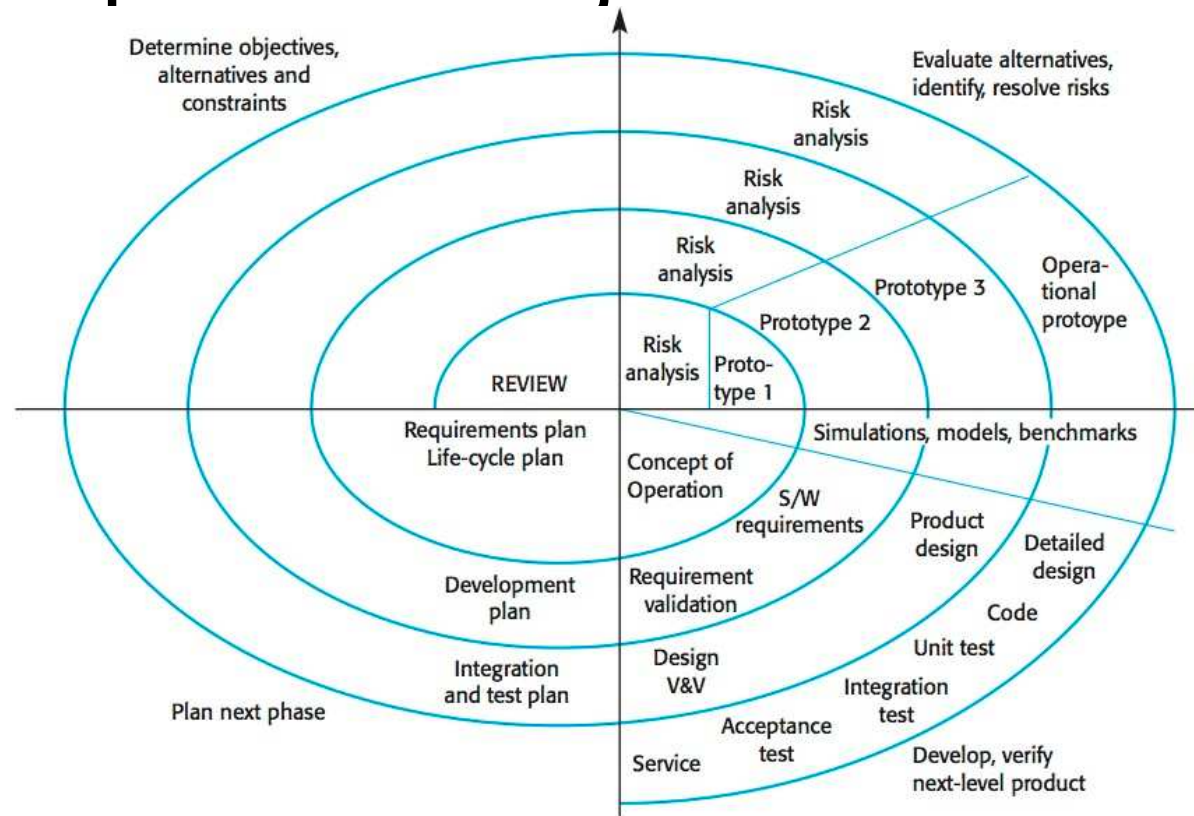
- Arrows show a sequential flow from **Definiálás** through the development phases to **Működés** (Operation).
- Feedback loops (indicated by arrows pointing back) exist from **Működés** to **Definiálás**, from **Implementálás** to **Implementáció tervezése**, and from **Implementáció tervezése** to **rendszer tervezés**.
- The **Prototípusok készítése** phase includes a loop labeled **1** and **2**, indicating iterative development.
- A large box labeled **Szimuláció, modellek, benchmarks** (Simulation, models, benchmarks) spans across the development and evaluation phases, with arrows connecting it to **Definiálás**, **Tervezés**, and **Implementálás**.

**Final Phases:**

- Befejezés** (Completion) follows the operation phase.
- Működés** (Operation) is the final state of the system.

# Inkrementális modellek

## Boehm spirál modellje



<http://iansommerville.com/software-engineering-book/web/spiral-model/>



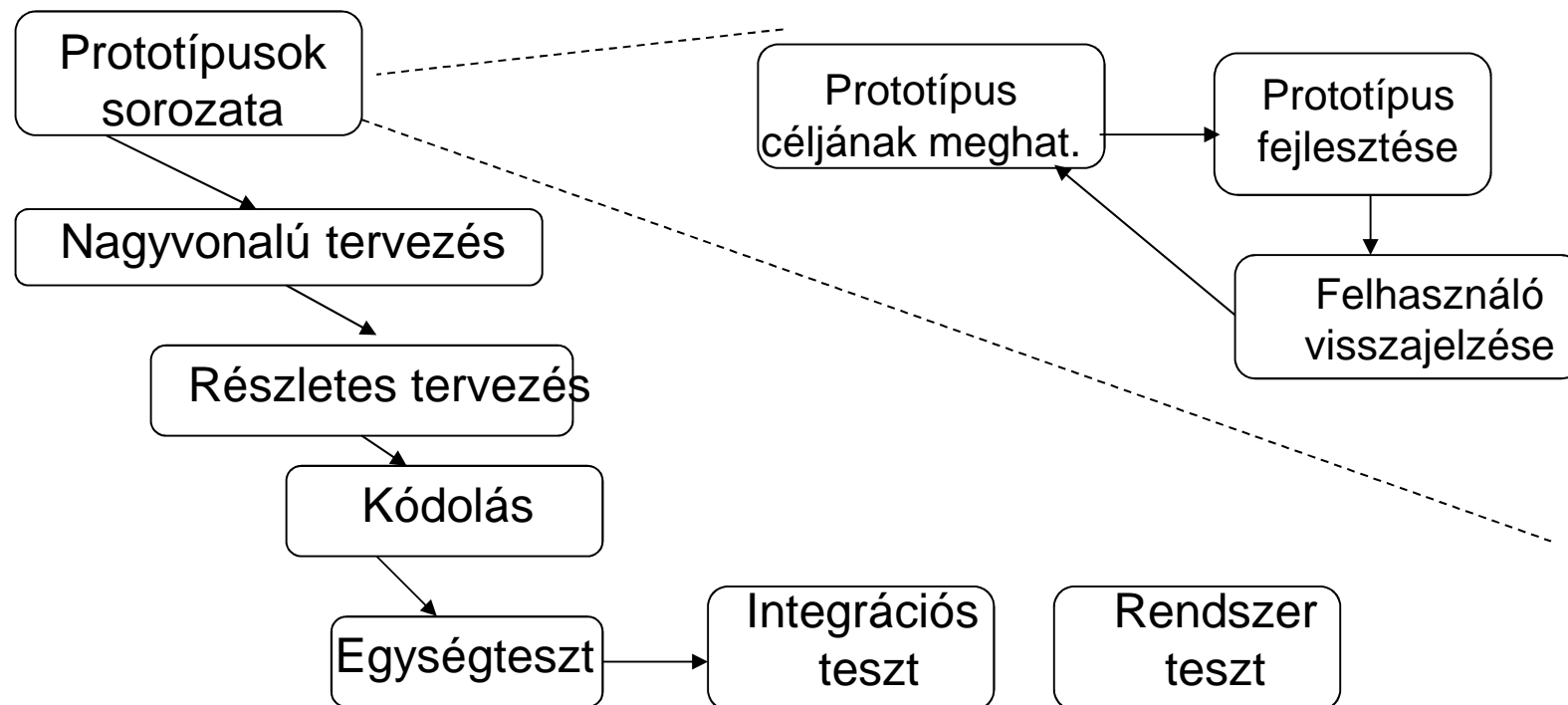
# Iteratív modellek

- Az iteratív modellek ciklikusak; előbb a követelményeknek egy kisebb (al)csoportját implementáljuk egy előre meghatározott időintervallum alatt – melyet iterációnak nevezünk.
  - Iteratív modellek tipikusan az **Agilis megközelítések** (Scrum, Kanban, XP, Lean) – lásd később, ebben az előadásban.

# Iteratív modellek

## ■ Evolúciós fejlesztés („rapid prototyping”)

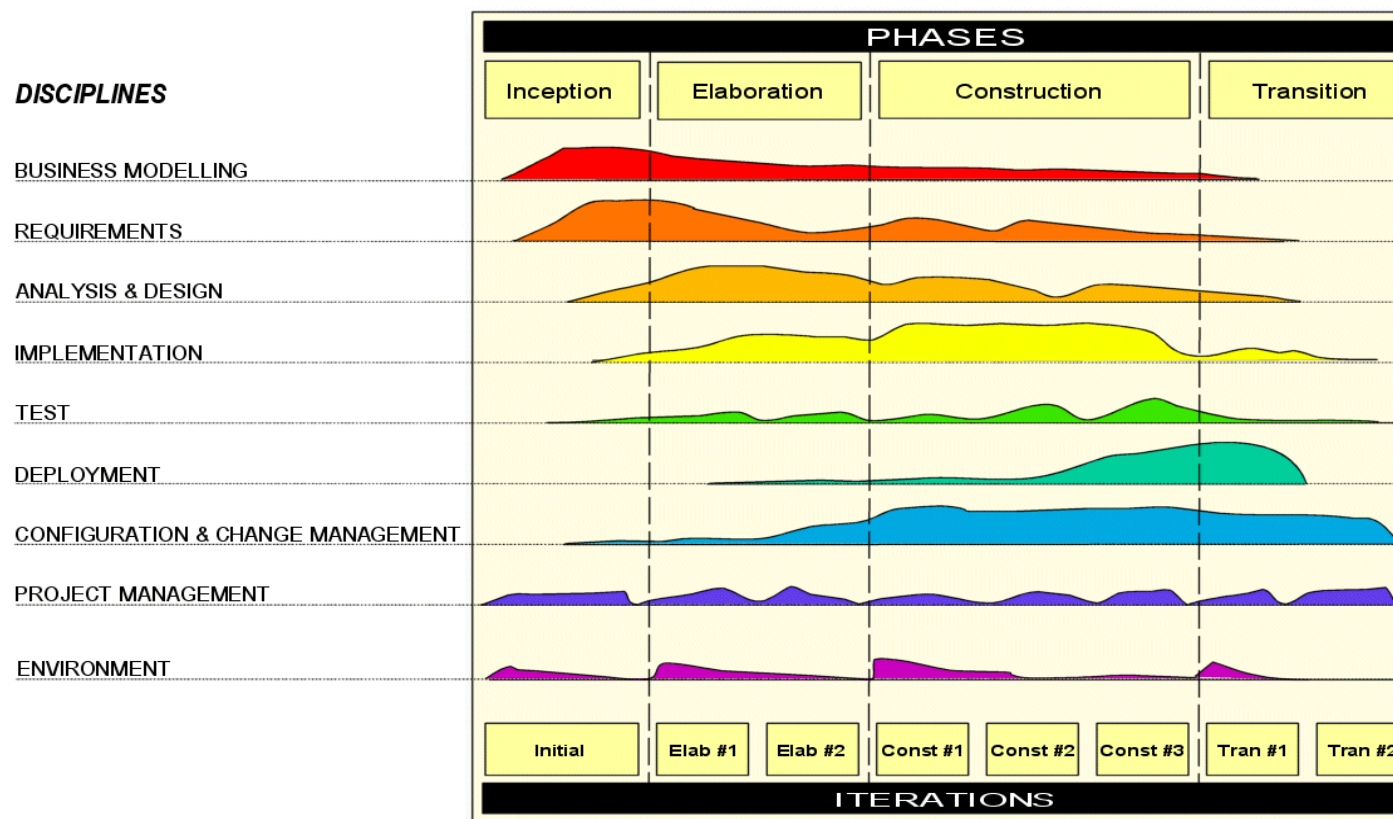
- Csak az első build-et definiálják, majd, a felhasználó prioritásainak alapján ezt egészítik ki újabb elemekkel



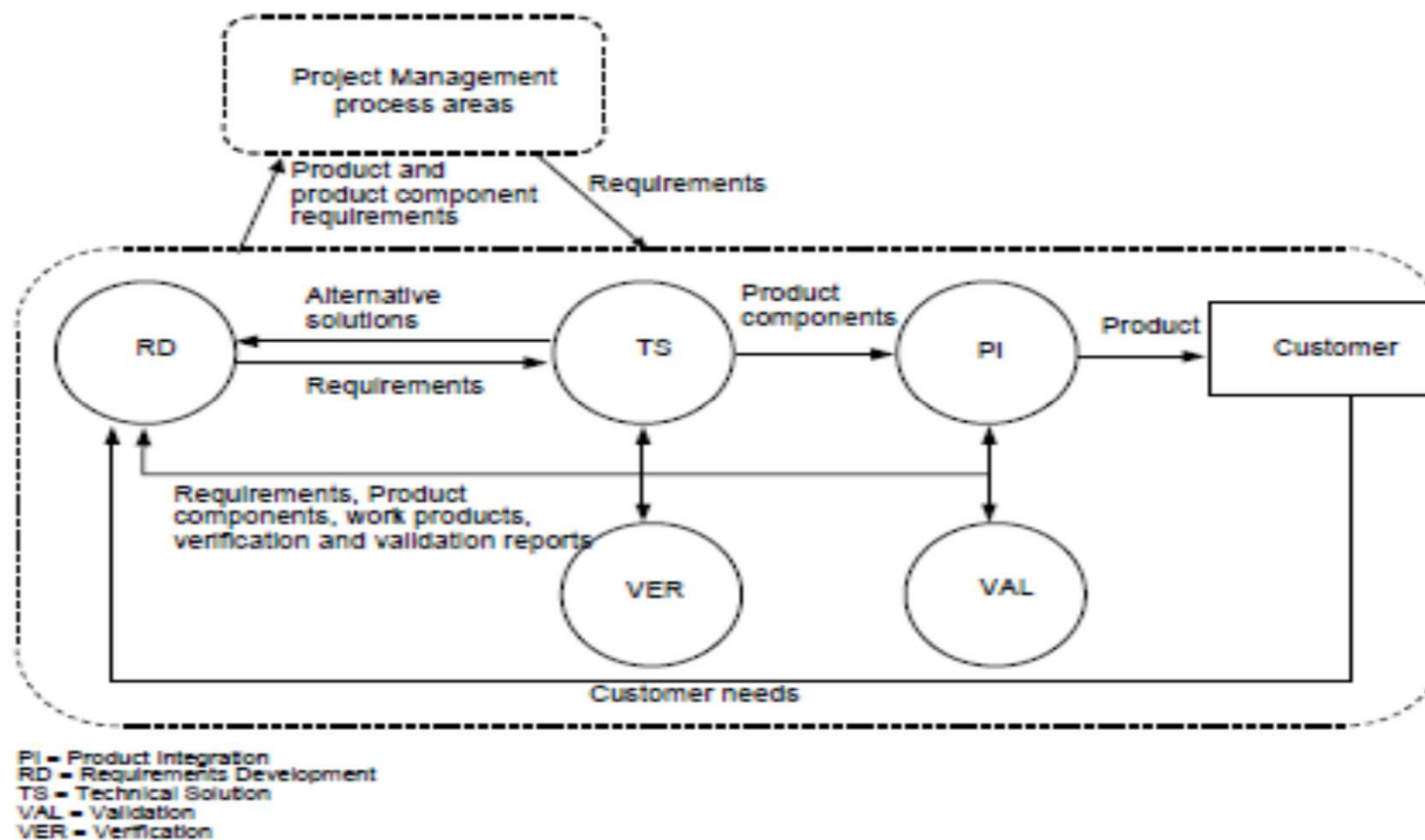


# Egyéb életciklus modellek / módszertanok

RUP: iteratív, inkrementális, use-case vezérelt  
(támogatja az OO fejlesztést)



# Szoftvermérnöki folyamatok a CMMI-ben



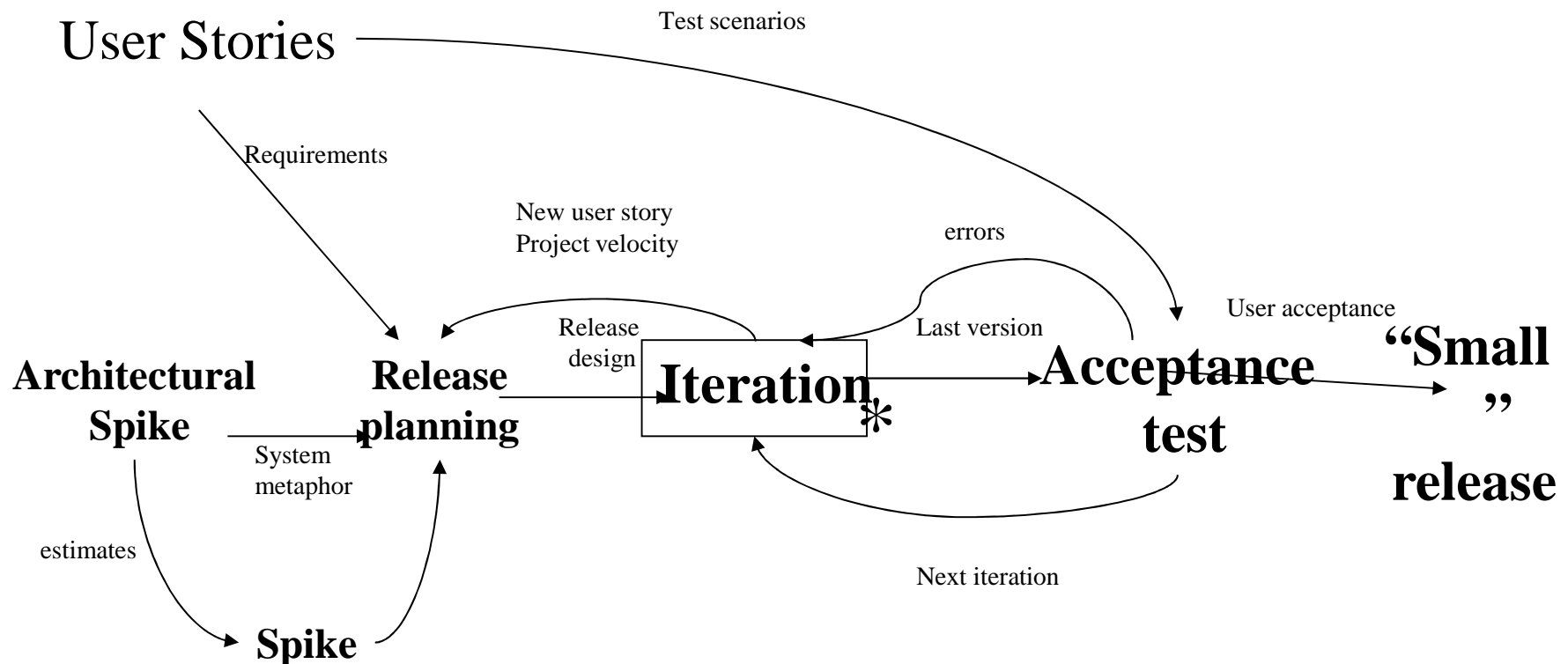


# Extreme programming

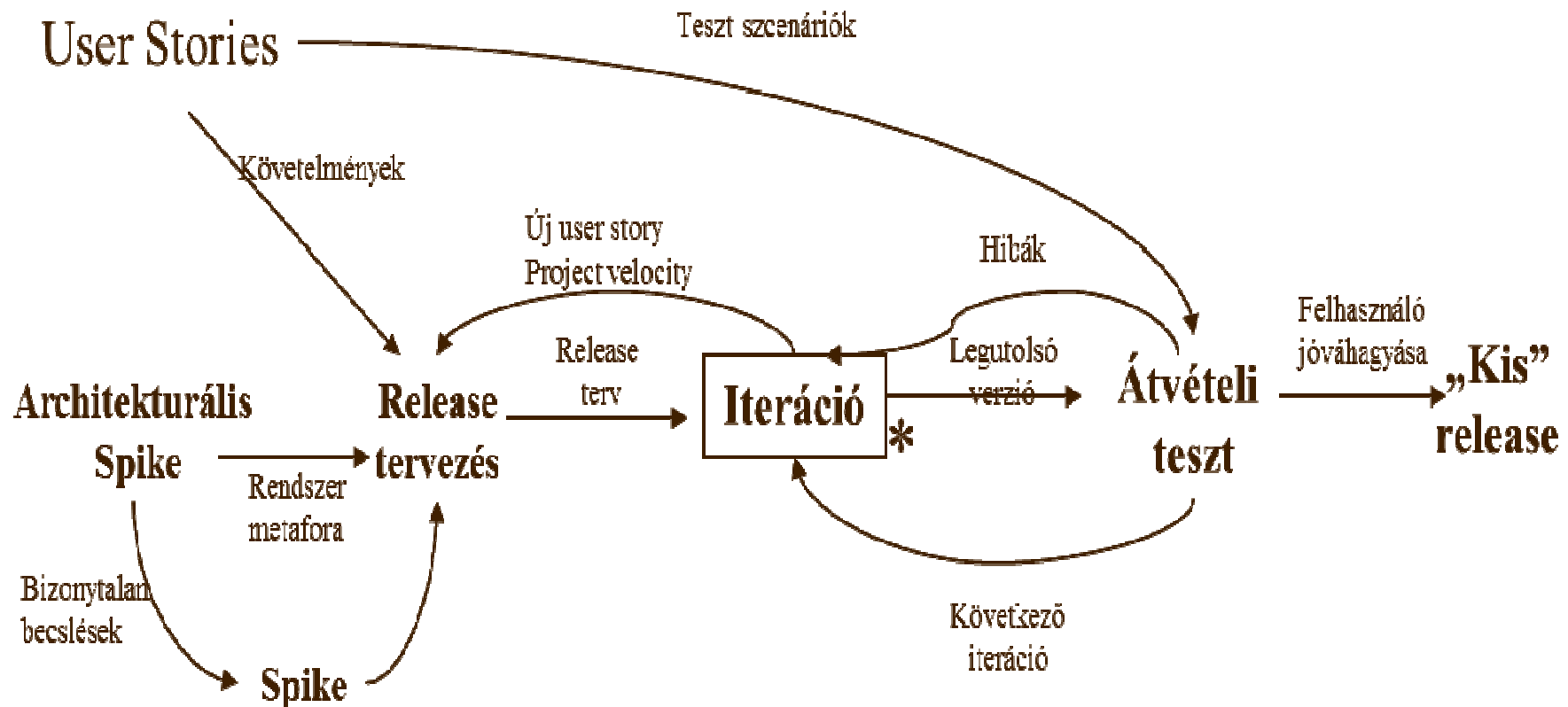
- Gyors, változékony, rugalmas projektek módszertana
- Könnyedén és rugalmasan kezeli a változásokat
- A következő tényezőkre épít:
  - Egyszerűség
  - Kommunikáció
  - Visszajelzés
  - Bátorság (új dolgok kipróbálásában)
- Un. „user story”-kra épít
- Un. „Spike”-okat (megoldásokat) használ ismétlődő feladatokra
- A tervek elkészülése után rövid időtartalommal, iteratív módon történik a fejlesztés
- Ha a teszt elfogadásra került, kész az adott verzió
- Együtt van ügyfél, menedzser és programozó
  - **Megj: Életciklus modell, vagy programozási módszer? Mindkettő lehet. Itt azért említjük, mert a szoftverfejlesztéshez kapcsolódó folyamatokat ír le, sorrendet javasol a végrehajtásukra.**



# Extreme programming



# Extreme programming



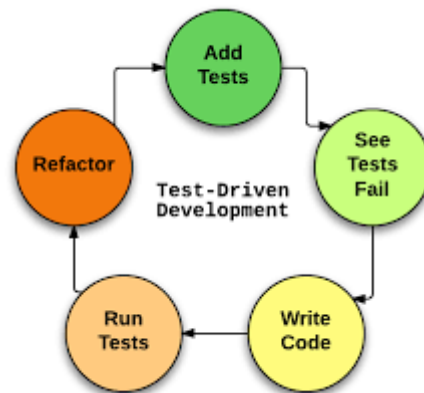
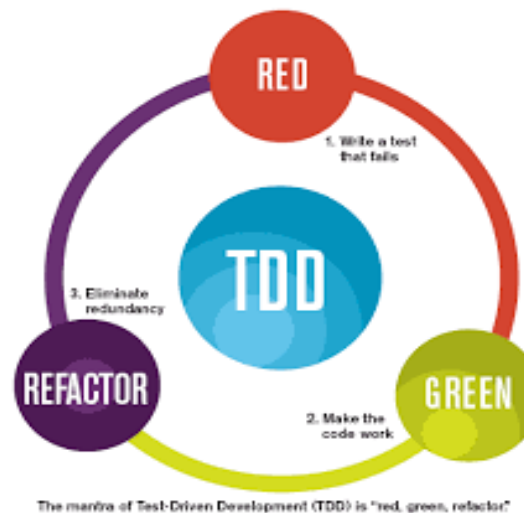
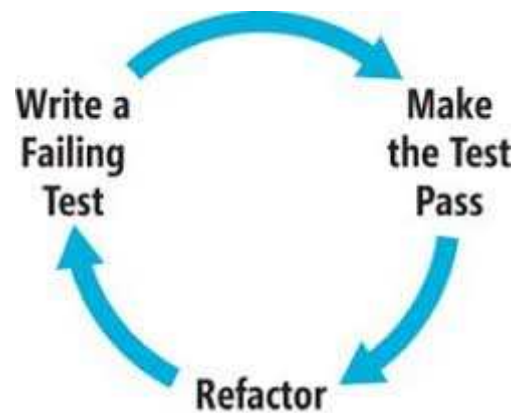


# Tesztevezérelt fejlesztés / Test Driven Development (TDD)

- A TDD-t általában az extrém programozásban és az agilis megközelítésekben alkalmazzák
  - Lásd 7. előadás
    - A programozó először a tesztekét írja meg
    - A tesztek előbb sikertelenek lesznek
    - Fokozatosan elkészül a tesztekhez tartozó kód
    - A tesztek újabb elemekkel egészítik ki
    - Megj: Életciklus modell, vagy tesztelési módszer? Mindkettő lehet. Itt azért említjük, mert a szoftverfejlesztéshez kapcsolódó folyamat sorrendet javasol a végrehajtásukra.

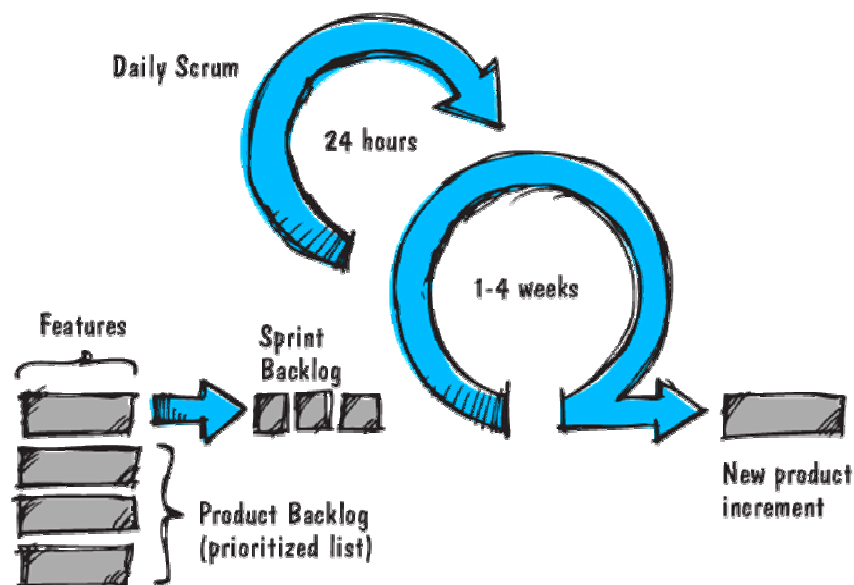


# Test Driven Development



# Scrum

- Gyakran emlegetik életciklus modellként; tulajdonképpen nem (csak) az!
- Sikeres agilis eszköz
  - További részletek a 11. előadásban
- Alapvetően egy menedzsment eszköz, melynek vannak a szoftverfejlesztéshez kapcsolódó elemei
- Itt említjük, mert ajánl a szoftverfejlesztésben alkalmazható életciklus modellt is





# Szoftver élelciklus modellek

- Mindegyik szoftverfejlesztési élelciklus modell (többé-kevésbé) ugyanazokból a folyamatokból épül fel
- A teljes „menü”, melyekből a folyamatokat kiválaszthatjuk, szabvánnyá vált
- ISO /IEC 12207: Information Technology - Software life cycle processes
  - *The international standard ISO 12207 is a globally accepted standard for software lifecycle processes. Though not suited for the direct application in a concrete project, it offers a frame that national standards or corresponding process details to be integrated in order to achieve a standard that can be used in an actual case. Furthermore, the ISO 12207 standard includes definitions that can be applied as a basis for a common terminology, even in national standards.*
- „This is the Process for Building Software In the 21st Century!”
  - <http://www.12207.com/>
  - <http://www.12207.com/12207-news.html>
- Teljesen kompatibilis a V-modellel



# Szoftver élelciklus modellek

- ISO/IEC 12207:2008 : Systems and software engineering -  
- Software life cycle processes.

<https://www.iso.org/standard/21208.html>

- Most (2017 őszén), fejlesztési fázisban egy újabb verzió:

- ☐ ISO/IEC/IEEE FDIS 12207: Systems and software engineering –  
Software life cycle processes

- ☐ <https://www.iso.org/standard/63712.html>

Korábban:

- ☐ [ISO/IEC 12207:1995](#)

Information technology -- Software life cycle processes

- ☐ [ISO/IEC 12207:1995/Amd 1:2002](#)



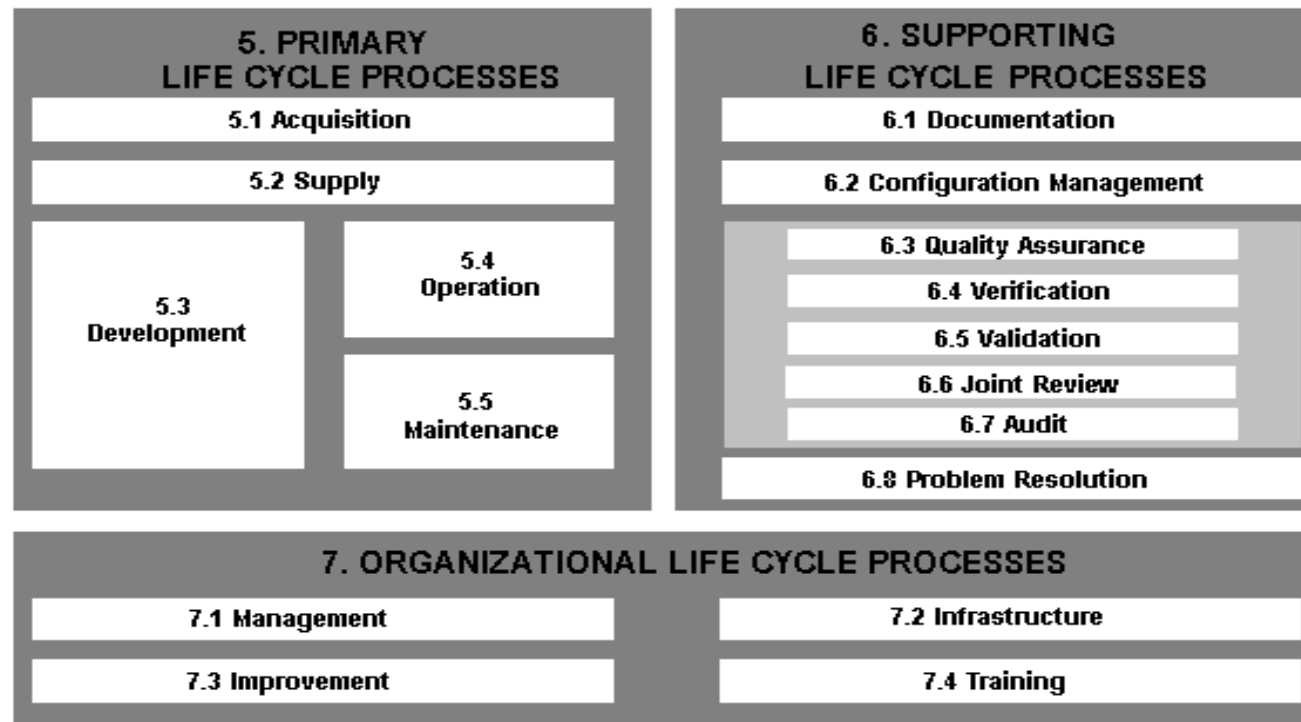
# Szoftver életciklus folyamatok

- Korábbi, magyar verzió:
- MSZ ISO/IEC 12207, 2000. május
  - Informatika. Szoftveréletciklus - folyamatok
- Egységes fogalmi keretet hoz létre a szoftveréletciklus-folyamatokra olyan jól meghatározott terminológiát használva, amely a szoftveripar számára kiindulásul szolgálhat. Olyan folyamatokat, tevékenységeket és feladatokat tartalmaz, amelyek szoftvert tartalmazó rendszerek, különálló szoftvertermékek és szoftverszolgáltatások beszerzése, valamint szoftvertermékek szállítása, fejlesztése, üzemeltetése és karbantartása során alkalmazandók. A „szoftver” kifejezés kiterjed a förmver szoftverrészére is.
  - Olyan folyamatot is tartalmaz, amelyet a szoftveréletciklus-folyamatok létrehozására, ellenőrzésére és javítására lehet használni.
  - Alkalmazható rendszerek, valamint szoftvertermékek és szoftverszolgáltatások beszerzésénél, szoftvertermékek és förmver szoftverrészének szállítása, fejlesztése, üzemeltetése és karbantartása során, függetlenül attól, hogy azt a szervezeten belül vagy kívül hajtják végre



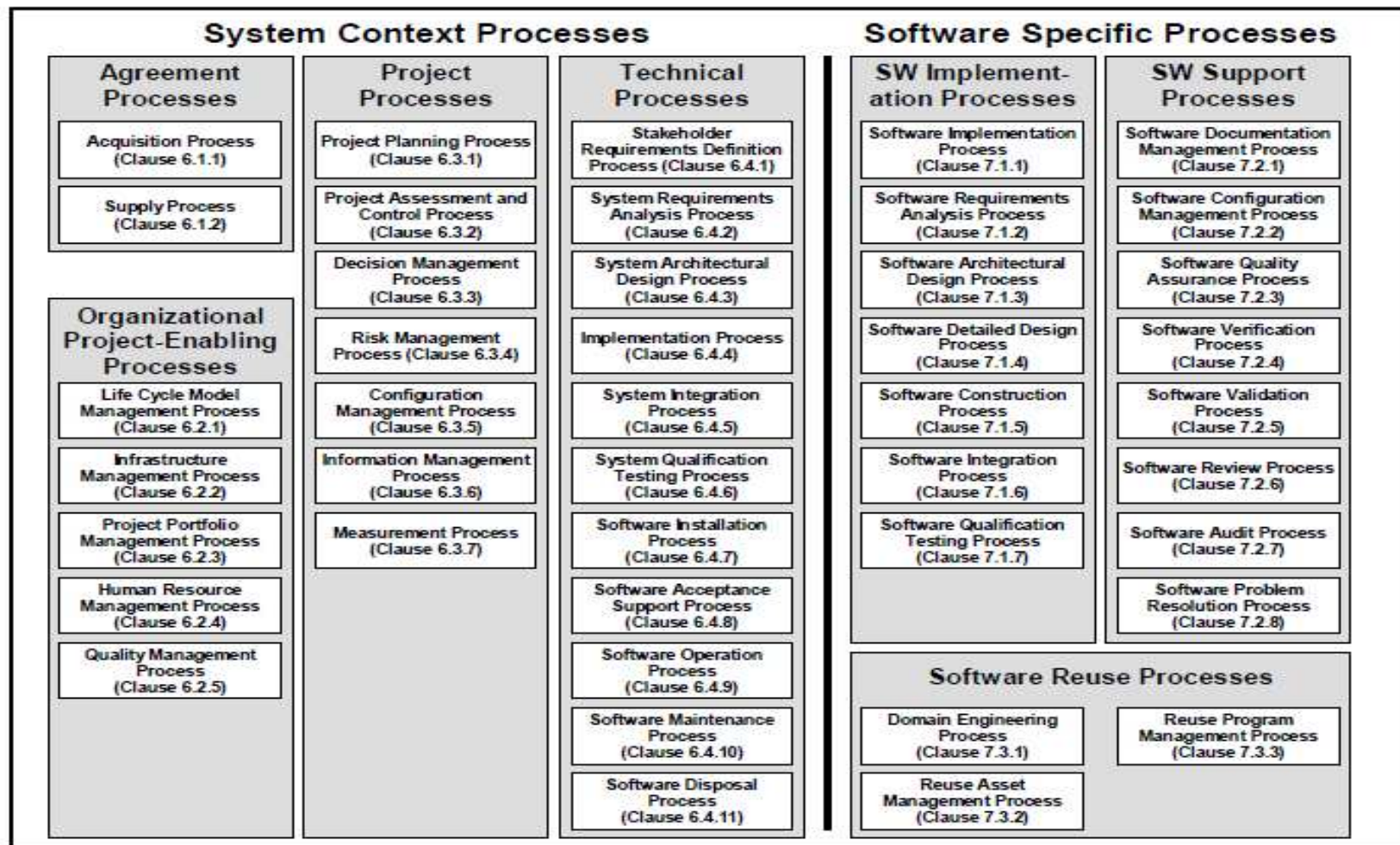
# Szoftver élelciklus folyamatok

- Az ISO 12207 szerint a következő folyamatok lehetségesek a szoftverfejlesztés során:



# Az ISO 12207

## ■ ISO/IEC 12207: 2008. Systems and software engineering- Software life cycle



2017.10.05.

Figure 1 — Life Cycle Process groups



# Az ISO 12207

- Legújabb, Draft verzió:
  - Kiadva: 2016.09.21 (szavazás kezdete)  
2016.12.13. (szavazás vége)
    - Még jobban hangsúlyozásra kerül a „rendszer” fogalma
  - Testreszabási útmutatók
    - Harmonizálják az ISO/IEC 33004 szabvánnyal

## ISO/IEC 12207/15288:2007 Process Constructs

Processes require a purpose, and outcome. All processes have at least one activity. The processes, with their statements of purpose and outcomes, constitute a Process Reference Model (PRM). PRM is given in Annex B.

Activities are constructs for grouping together related tasks. The activities provide a means to look at related tasks within the process to improve understanding and communication of the process. If an activity is cohesive enough, it can be converted to a (lower level) process by defining a purpose and a set of outcomes.

A task is a detailed provision for implementation of a process. It may be a requirement ("shall"), a recommendation ("should"), or a permission ("may").

Notes are used when there is a need for explanatory information to better describe the intent or mechanics of a process. Notes provide insight regarding potential implementation or areas of applicability such as lists, examples and other considerations.

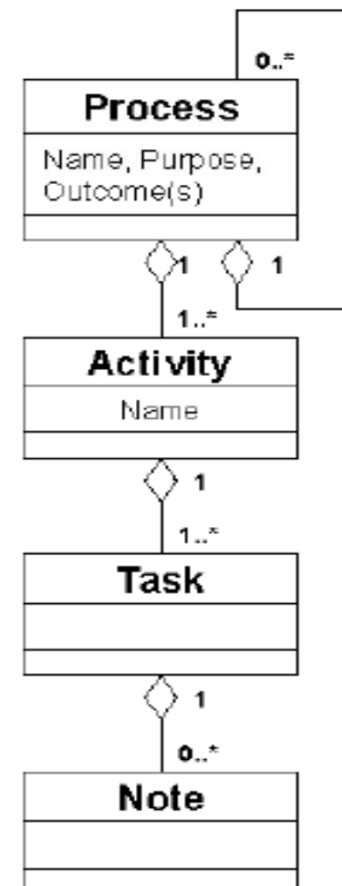


Figure C.1 — ISO/IEC 12207/15288 Process Constructs



# Szoftver életciklus folyamatok

- Két felet érintő helyzetekben történő használatra tervezték, de alkalmazható akkor is, ha a két fél ugyanabból a szervezetből való. Megállapodás (hivatalos szerződés / nem hivatalos megállapodás) szükséges.
- E szabványnak való megfelelés azt jelenti, hogy végrehajtották az összes olyan folyamatot, tevékenységet és feladatot, amit a szoftverprojekt céljaira az illesztési folyamat segítségével ebből a szabványból kiválasztottak.



# Hagyományos és agilis megközelítések

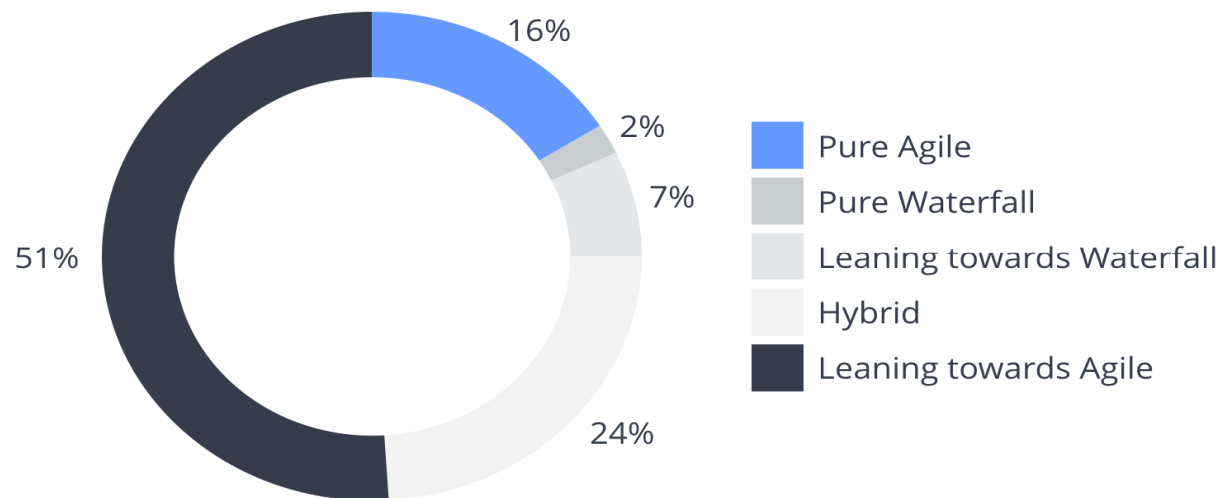
## ■ Hagyományos szoftverfejlesztés

- Általában egy sor, egymás után végrehajtott lépésből áll
- Több dokumentációt igényel
- „Nehézkesebb”, formálisabb
- Feltétel, hogy a felhasználónak legyen részletes elképzelése a készítendő szoftverről, és a követelmények kerüljenek részletes megfogalmazásra

## ■ Agilis szoftverfejlesztés:

- A szoftverfejlesztési módszerek csoportja, amely az iteratív fejlesztési modellen alapul, mind a követelmények, mind a megoldások az önszerveződő, széleskörű feladatokkal rendelkező csapatok együttműködése által jönnek létre
  - (Az ISTQB Glossary alapján)
- Kevésbé formális
- A felhasználó részt vesz a fejlesztésben, a követelmények a fejlesztés elején még nem teljesen tisztázottak

# Hagyományos és agilis megközelítések



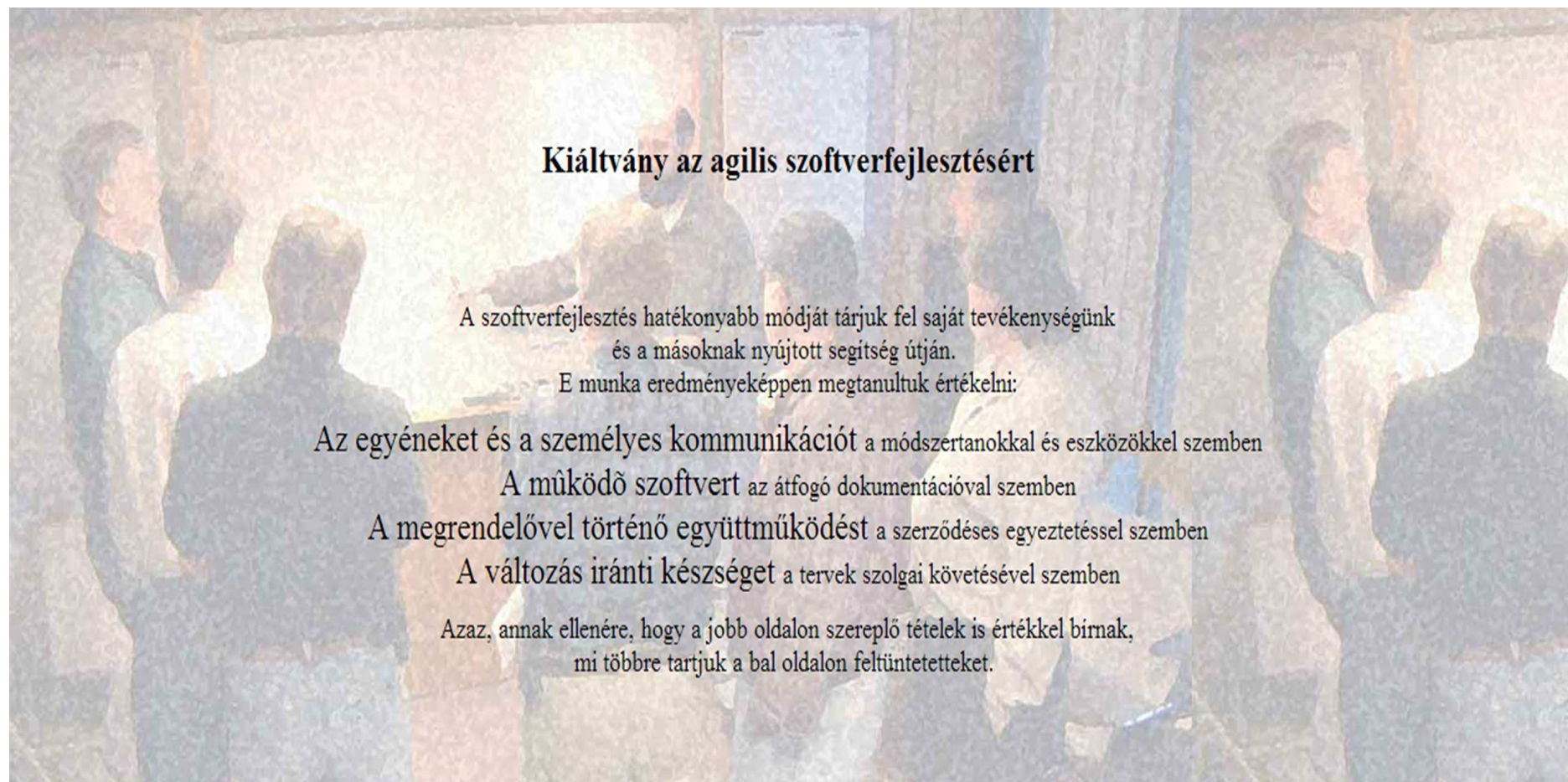
<https://techbeacon.com/survey-agile-new-norm> (2017)



# Agilitás dióhéjban

„On February 11-13, 2001, at The Lodge at Snowbird ski resort in the Wasatch mountains of Utah, seventeen people met to talk, ski, relax, and try to find common ground—and of course, to eat.

What emerged was the Agile ‘Software Development’ Manifesto.”



## Kiáltvány az agilis szoftverfejlesztésért

A szoftverfejlesztés hatékonyabb módját tárjuk fel saját tevékenységünk  
és a másoknak nyújtott segítség útján.

E munka eredményeképpen megtanultuk értékelni:

Az egyéneket és a személyes kommunikációt a módszertanokkal és eszközökkel szemben

A működő szoftvert az átfogó dokumentációval szemben

A megrendelővel történő együttműködést a szerződéses egyeztetéssel szemben

A változás iránti készséget a tervek szolgái követésével szemben

Azaz, annak ellenére, hogy a jobb oldalon szereplő tételek is értékkel bírnak,  
mi többre tartjuk a bal oldalon feltüntetetteket.

**Balla K.**

2017.10.05.

<https://www.agilealliance.org/agile101/the-agile-manifesto/>

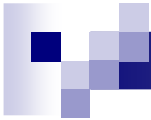
**40**





# 12 agilis alapelv

- 1) Legfontosabbnak azt tartjuk, hogy a vevőnk elégedett legyen, mert értékes szoftvert szállítunk neki hamar és folyamatosan.
- 2) Elfogadjuk, hogy a követelmények változhatnak akár a fejlesztés vége felé is. Az agilis módszertanok befogadják a változást a megrendelő versenyképességének érdekében.
- 3) Gyakran szállítsunk működő szoftvert, pár hetes és hónapos időközönként, lehetőleg a rövidebb periódust választva.
- 4) A megrendelők, üzleti szakemberek és a szoftverfejlesztők dolgozzanak együtt minden nap a teljes projekt során.
- 5) Építsük motivált egyénekre a projekteket. Teremtsük meg nekik a számukra szükséges környezetet és támogatást, és bízzunk bennük, hogy elvégzik a munkát.
- 6) A személyes beszélgetés az információ átadásának leghatásosabb és hatékonyabb módja a fejlesztő csapaton belül.
- 7) Az előrehaladás elsődleges mércéje a működő szoftver.
- 8) Az agilis módszertanok elősegítik a fenntartható fejlesztést. A szponzoroknak, fejlesztőknek, felhasználóknak korlátlan ideig képesnek kell lenniük az egyenletes sebesség megőrzésére.
- 9) A folyamatos figyelem a technikai kiválóságra és a jó tervezésre fokozza az agilitást.
- 10) Az egyszerűség – az el nem végzett munkamennyiség maximalizálásának művészete – alapvető érték.
- 11) A legjobb architektúrák, követelmények és rendszertervek az önszerveződő csapatmunkából alakulnak ki.
- 12) A fejlesztői csapat rendszeres időközönként megfontolja, hogyan válhatna hatékonyabbá és ennek megfelelően változtatja viselkedését.



# Agilitás dióhéjban

- Minden projekt, amelyik a 12 előbbi alapelveket alkalmazza, agilisnak tekinthető.
- Az agilitás egy jól kidolgozott gondolatrendszer (filozófia), amelyik lehetővé teszi termékek lépésről lépésre történő, inkrementális fejlesztését, a terméket kisebb részekben átadva. Az agilis gondolkodásmód időközpontú és iteratív.
  - „Agilitás”: latin eredetű szó. Eredeti jelentése: serénység, ügyesség, élelmesség, életrevalóság, gyorsaság, mozgékonyaság



# Agilis gondolkodásmód

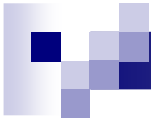


- Fogadjuk el a változásokat!
- A szoftver folyamatosan fejlődik; a felhasználó állandó visszajelzést ad.
- Folyamatos az átadás (Continuous delivery)
- Érték-vezérelt
- Minden átadásnál egy kis résszel egészül ki a szoftver. A hozzáadott részek növelik a szoftver értékét.

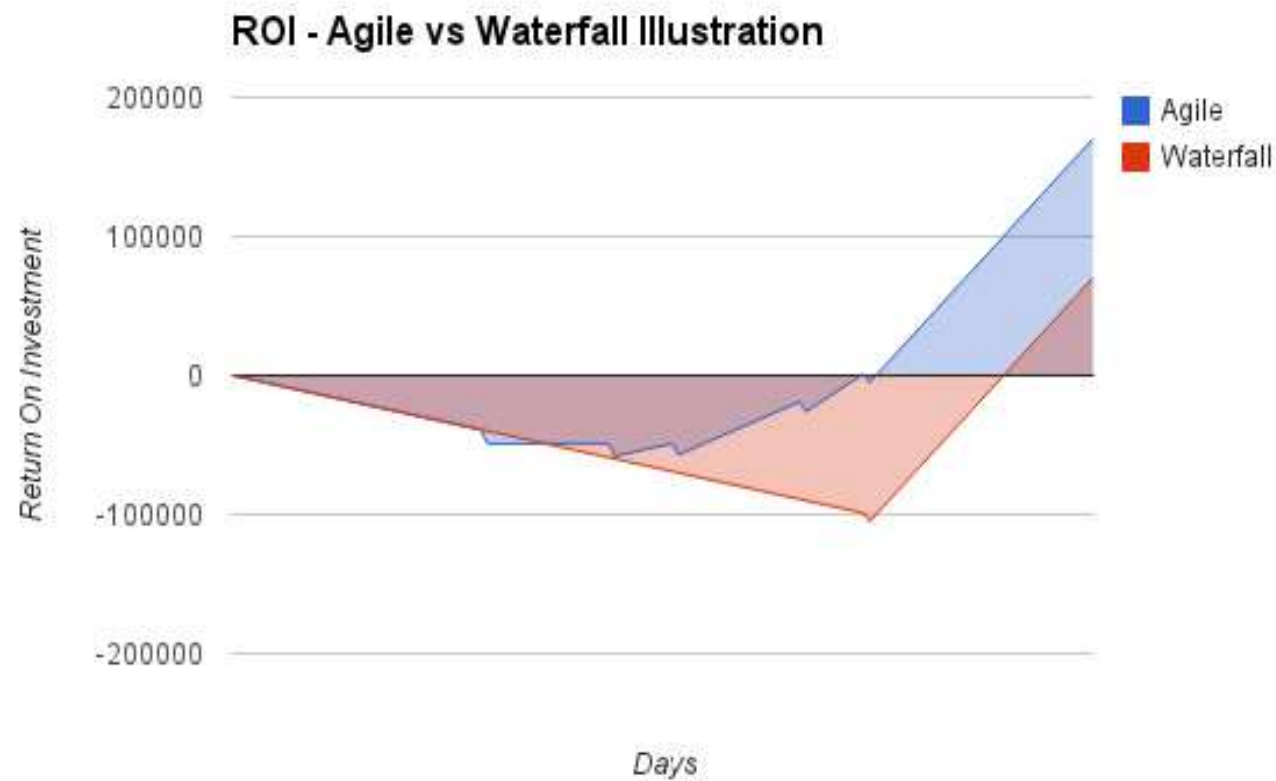


# Agilis és vízésés modell

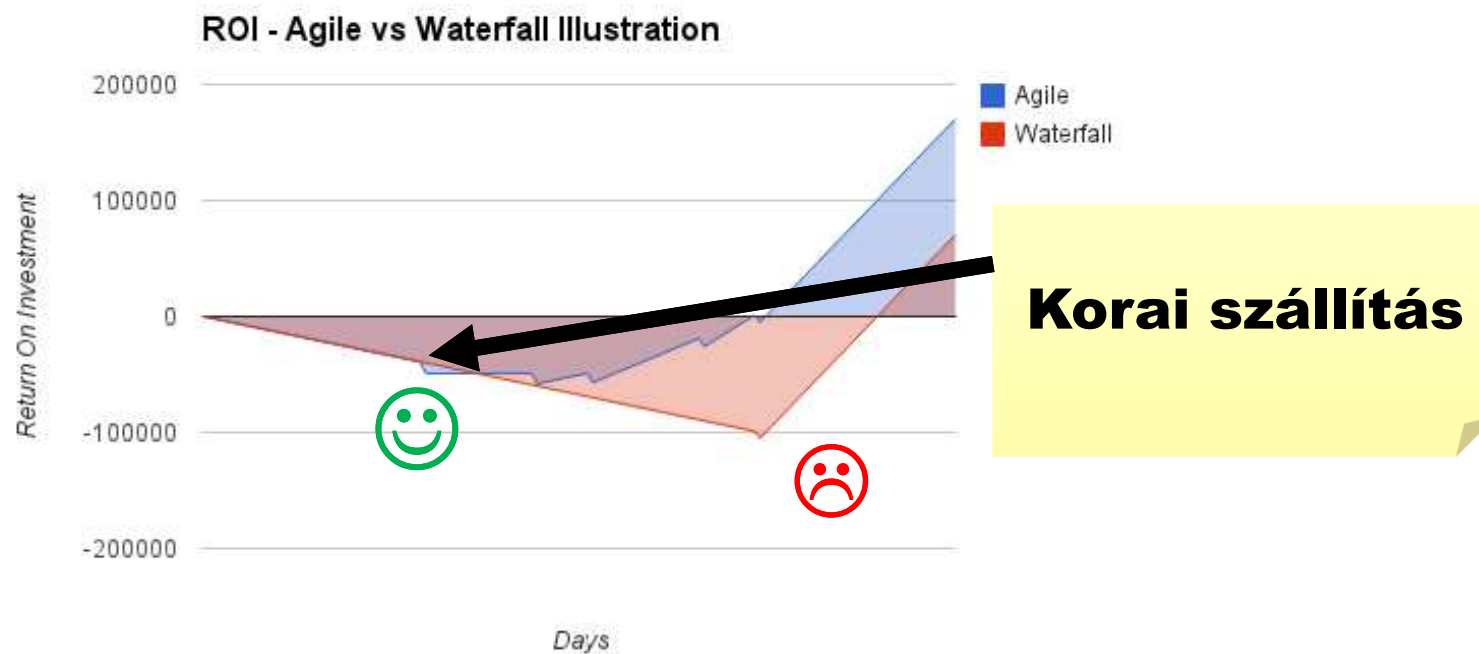
Agilis	Vízésés
Folyamatos visszajelzés	Feedback at the end of the development
Minimális dokumentáció	Részletes dokumentáció
Reagálás a változásokra	A terv követése
Minden iteráció minden munkafázist tartalmaz	A fázisokat szigorúan csak egymás után lehet végrehajtani
Nem ismert minden specifikáció	Minden funkciót előre specifikálni kell
A fázisok nem különállóak. Az egész csapat ismer minden feladatot. A folyamatok párhuzamosan futhatnak.	A különböző fázisokban különböző szakemberek vesznek részt, akik különböző típusú tudással rendelkeznek



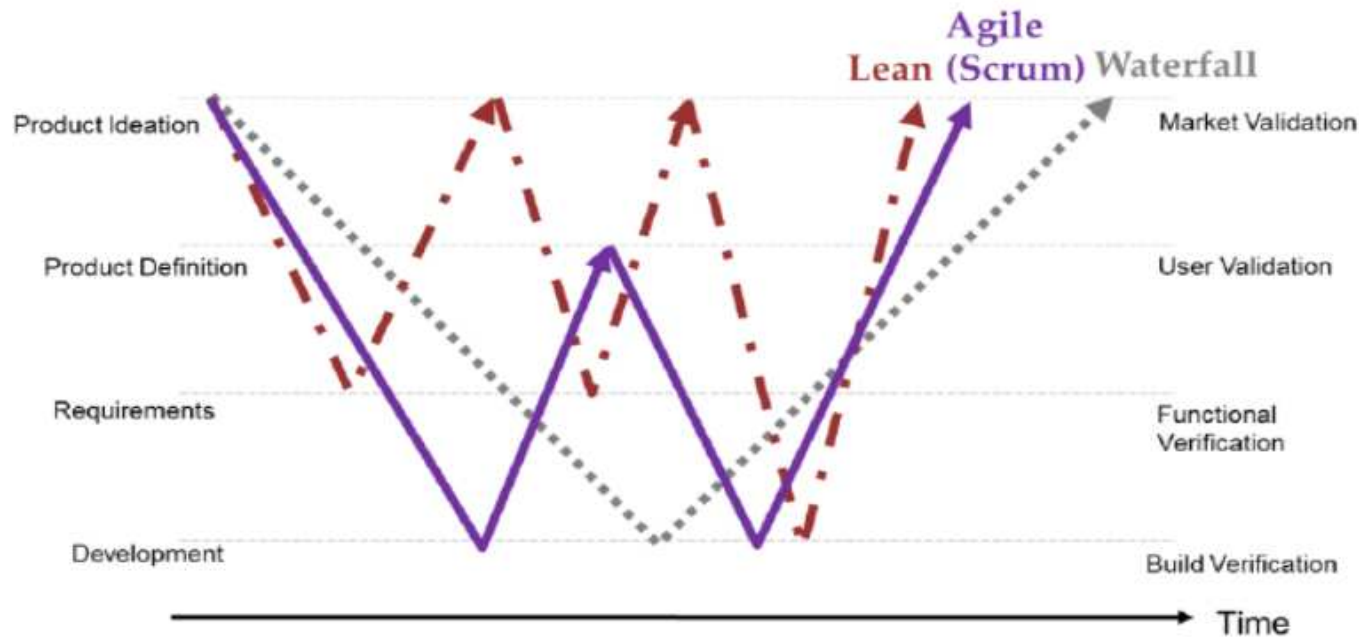
# Agilis és vízésés modell



# Agilis és vízésés modell



# Hagyományos és agilis megközelítések



12



<https://www.cloudteams.eu/projects/>



# Agilitás dióhéjban

- Egy agilis cég általában nagyon rugalmas, könnyen alkalmazkodik a változásokhoz, kevesebbet iterál és gyorsabban implementál, felismeri az új lehetőségeket. Gyors a döntéshozatali mechanizmus, rugalmas a szervezeti struktúra, egyszerű a kommunikáció.
- 2015-ben kutatást folytattak 601 szoftverfejlesztési és IT szakember bevonásával; az eredmény azt mutatja, hogy manapság az agilitás a vezető menedzsment megközelítés.

<https://techbeacon.com/survey-agile-new-norm>



# Agilitás dióhéjban

## ■ Az agilis megközelítés előnyei



<https://techbeacon.com/survey-agile-new-norm>



# Mit lehet agilisan csinálni a szoftverfejlesztési projektekből?

## ■ Mindent! Lehet agilis:

### □ A szoftverfejlesztési módszertan

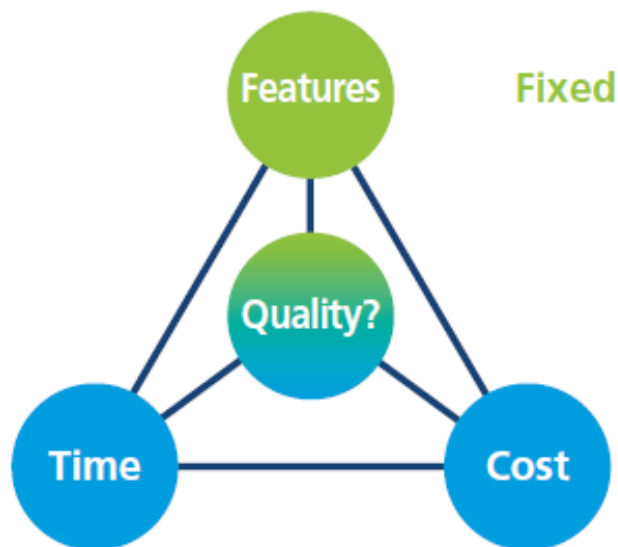
- .. És minden folyamata:
- Követelmények meghatározása
- Tervezés
- Kódolás
- Tesztelés
- ...

### □ A projektmenedzsment módszertan

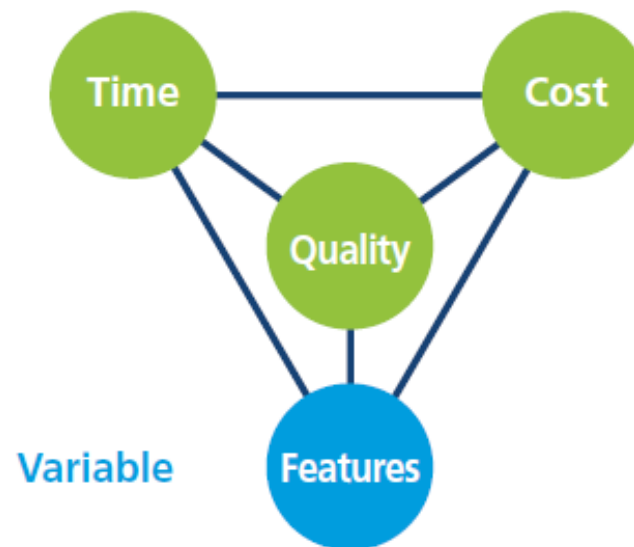
- Projekt tervezés
- Projekt követés és vezérlés
- változáskezelés

# Agilitás dióhéjban

Traditional Approach



Atern Approach



<https://www.agilebusiness.org/resources/ds-dm-handbooks/dsdm-atern-handbook-2008>

Atern is a proven 'battle hardened approach' and has been responsible for the successful delivery of innumerable projects around the world. Its provenance across both IT and non-IT contexts goes back to 1994 with substantial productivity gains independently verified by the UK Software Metrics Association.

# Agilitás dióhéjban

- Képzeljük az ügyfél helyébe magunkat

- ☐ Jó dolgok fognak így kisülni!

- **Visszajelzést kérünk**

- ☐ Rendszeresen

- ☐ Jó az irány?

- ☐ Az ügyfél is kell hozzá

- Alaptétel. Az ügyfél valamilyen formában elérhető.

- ☐ Skype?

- ☐ Adott órákban?

- ☐ **Személyesen**? „Orákulum”. – Ez nagyon jó. Szeretnénk.



*„Customer collaboration,  
over contract negoation”*

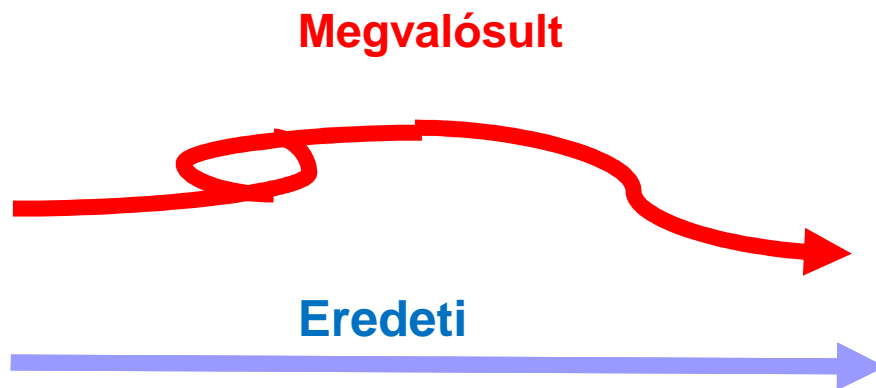
# Agilitás dióhéjban

## ■ Változtatunk – ha szükséges!



WISDOM

*„Responding to change,  
over following a plan”*

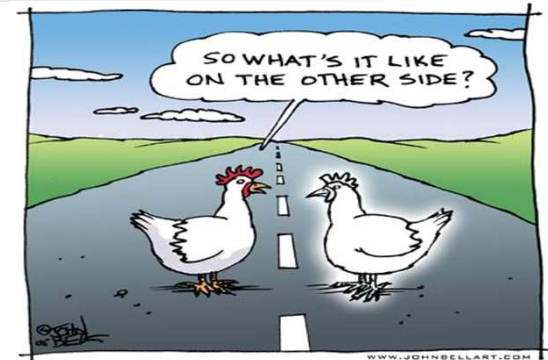


# Agilitás dióhéjban

- Másfajta munkavégzés
- **Nem való mindenkinek**

- Tipikusan:

- ☐ Kreatívabb
- ☐ Élénkebb kommunikáció
- ☐ Nagyobb mozgástér
- ☐ Kereszt-funkcionális csapatok
- ☐ Kevésbé monoton



- ☐ Több felelősség
- ☐ (Többet) kell gondolkodni
- ☐ Magas motiváció szükséges
- ☐ Önfegyelem szükséges
- ☐ Valódi csapatmunka szükséges

# Agilitás dióhéjban



## ■ Felelősség az agilis szoftverfejlesztésben

- ☐ Tipikusan nincs projektmenedzser
- ☐ Ha van is, nem a klasszikus értelemben
- ☐ A **csapat** vállalja el a feladatokat
- ☐ A **csapat** végzi el a munkát
- ☐ A **csapat** mutatja be az eredményeket
- ☐ Az **ügyfél** a csapattal **együtt** dolgozik

# Agilitás dióhéjban

## ■ Agilis szerződések

- Nem kerül minden rögzítésre
- Tipikusan valamit muszáj fixálni
- Alapvetően más
- Egy jó szerződés nem elég
- Sikeres szerződés – támogatja:

- A kollaborációt
- Az átlátszóságot/átláthatóságot (transparency)
- A bizalom kialakulását, fenntartását



*„Customer collaboration over contract negotiaton.”*



# Agilitás dióhéjban

## ■ Az agilis csapat

☐ Kereszt-funkcionális

☐ Önszerveződő

☐  $2 \sim 3 < \text{méret} < 20 \sim 25$

■ Túl kicsi: Legtöbb módszertan értelmét veszti

■ Túl nagy: Kommunikációs overhead

☐ Co-Location

■ A leghatékonyabb kommunikációs módszer az élőszó.

☐ Nagyobb csapatokra, szervezetekre:

■ Scaling-Agile

■ Felbontás

■ Scrum-of-Scrums

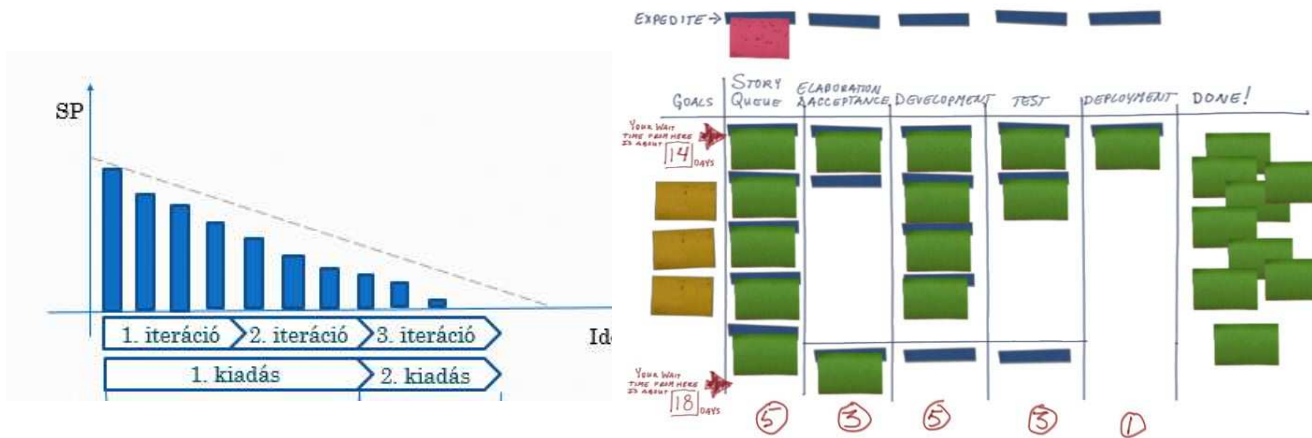
*„Build projects around motivated individuals.  
Give them the environment and support they need,  
and trust them to get the job done.”*

☐ Nagyban csinálni; nehézkes lehet!



# Agilitás dióhéjban

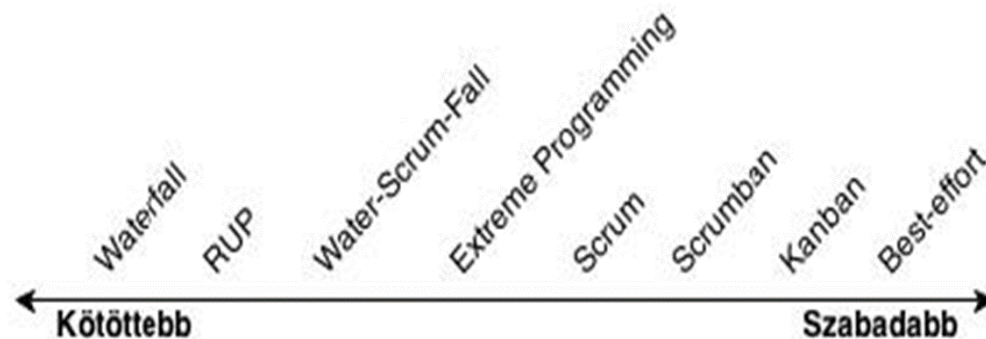
## ■ Vizualizáljunk, képesítsünk!



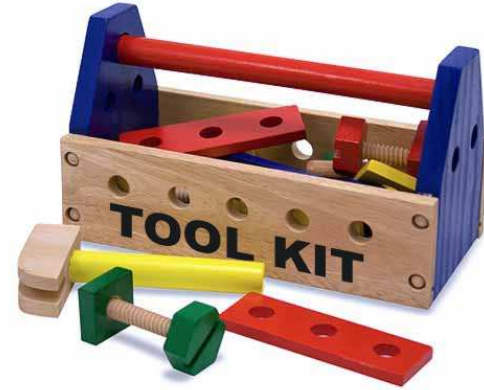
# Agilitás dióhéjban

## Agilis módszerek

- Adaptive Software Development
- Agile Modeling
- Agile Unified Process
- Crystal Methods (család)
- Disciplined Agile Delivery
- Domain Driven Design
- Dynamic Systems Development Method
- Extreme Programming
- Feature Driven Development
- Kanban
- Lean software development
- Scrum
- Scrum-ban
- Test Driven Development
- Water-Scrum-Fall
- + sok további?



# Agilitás dióhéjban



- Továbbra is kell gondolkodni. ☹ / ☺ **SŐT!**
- Nem egy végső, ultimate megoldás.
  - „No silver bullet!”
- Segíthet „elviselni” a bizonytalanságot
  - Együtt kell vele tudni élni.
  - Része az életünknek.
- Továbbra sincs itt a Kánaán
  - De hasznos eszközkészletnek bizonyulhat!



# Agilitás dióhéjban

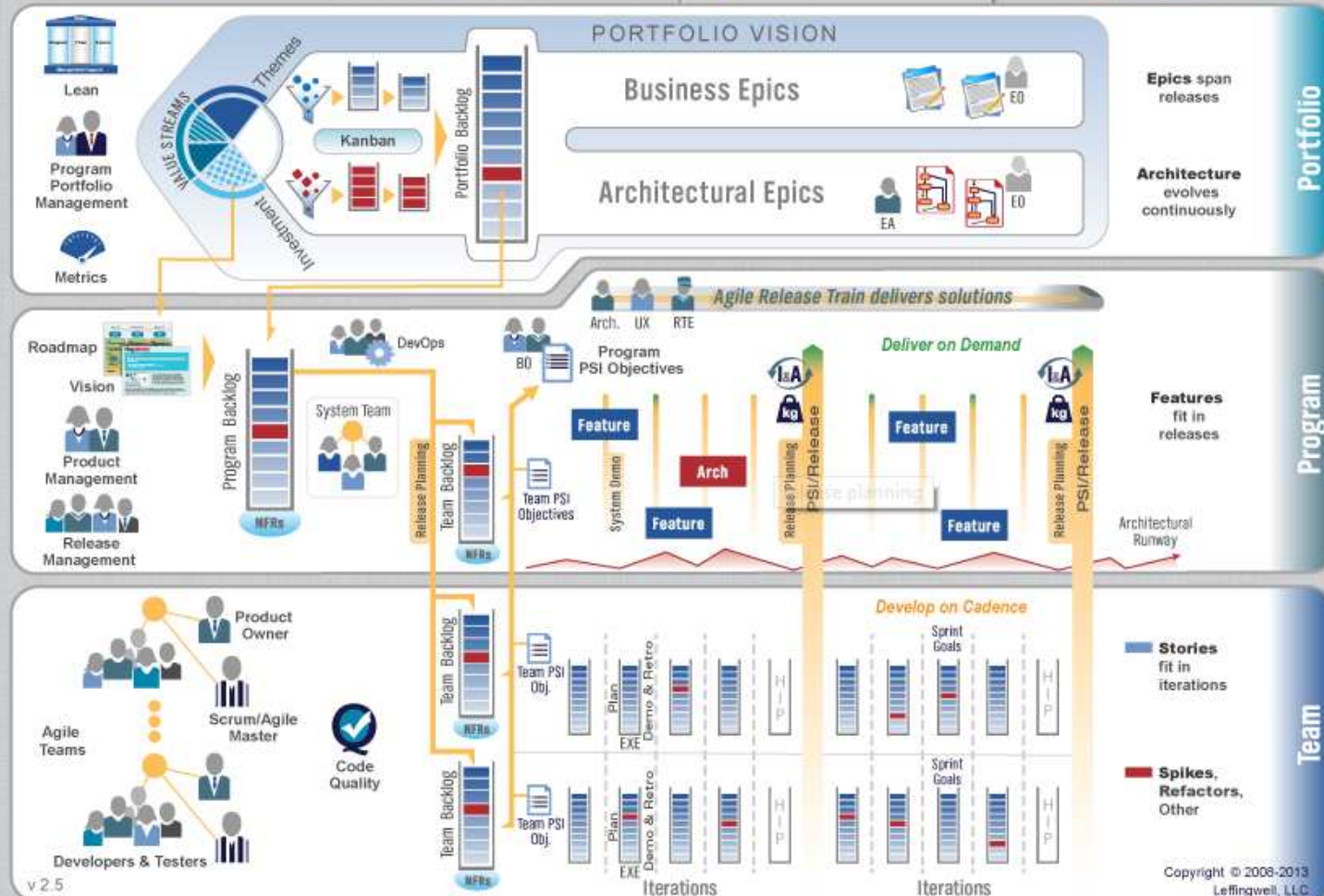
## ■ Scaled Agile Framework (TM)

- Lean-Agile
- Jól átgondolt, konzisztens megközelítés
- Nagyvállalati környezetben is használható!
- 3 szint:
  - Vízió (Epic)
  - Funkció (Higher level)
  - Feature (User Story)

# Scaled Agile Framework® Big Picture

CLICK ANY ICON  
for detailed information

Scaled Agile  
Framework





# Lean

- Egy vállalatsservezési, vállalatirányítási rendszer, amelynek célja, hogy a vállalat minél gazdaságosabban állítsa elő a termékeit, szolgáltatásait.
- A lean filozófiának a Toyota Motor Corporation Toyota Termelési Rendszere (TPS, Toyota Production System, újabb megközelítésben Thinking People System) képezi az alapját. A TPS az 1950-es években született, Taylor, Gilbreth, Smiles, Miles és Gantt munkássága és a Ford addig elért eredményei nyomán..
- A 2000-es években a Lean-t a szoftverfejlesztésre is adaptálták. Mary és Tom Poppendiecks összekapcsolták a 7 eredeti Lean alapelvet az agilis filozófiával.

□ Lean- eredeti jelentése: 'karcsú'





## 7 Lean alapelv

- Küszöböljük ki a veszteséget
- Hangsúlyozzuk a tanulást
- A lehető legkésőbbben döntsünk
- A lehető leggyorsabban adjuk át
- Erősítsük a csapatokat, legyen hatalmuk
- Építsük be az integritásra
- Lássuk a teljes rendszert

További részletek: <http://unipub.lib.uni-corvinus.hu/161/1/Losonci119.pdf>



# Lean és agilitás



**Philosophies:** Lean, Agile, etc.

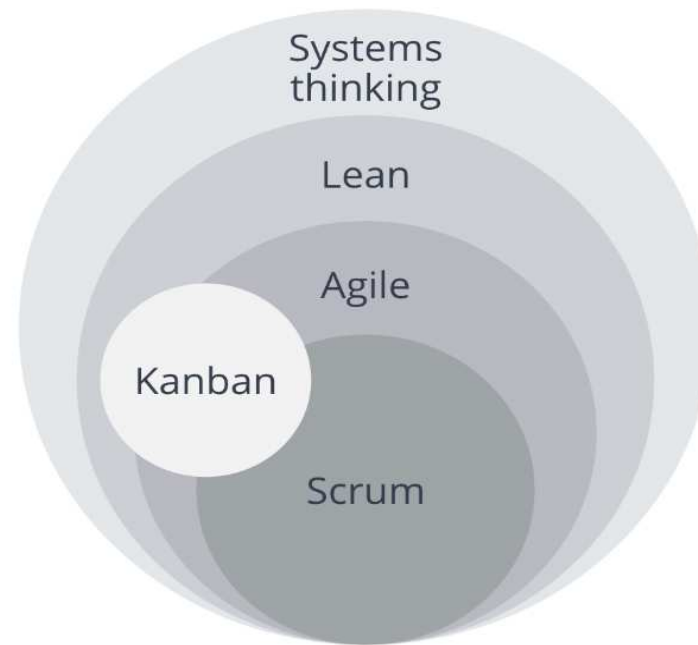
**Methodologies:** Scrum, Kanban, XP, TPS, etc.

**Tools:** sprints, boards, tests, cohort analysis, etc.

<https://realtimeboard.com/blog/choose-between-agile-lean-scrum-kanban/#.WdX9jjtx2QM>

NB.: Az „Agilis” és a „Scrum” nem  
összehasonlíthatók, mert más –  
más fogalmi szinten vannak!

# Agilis és Lean



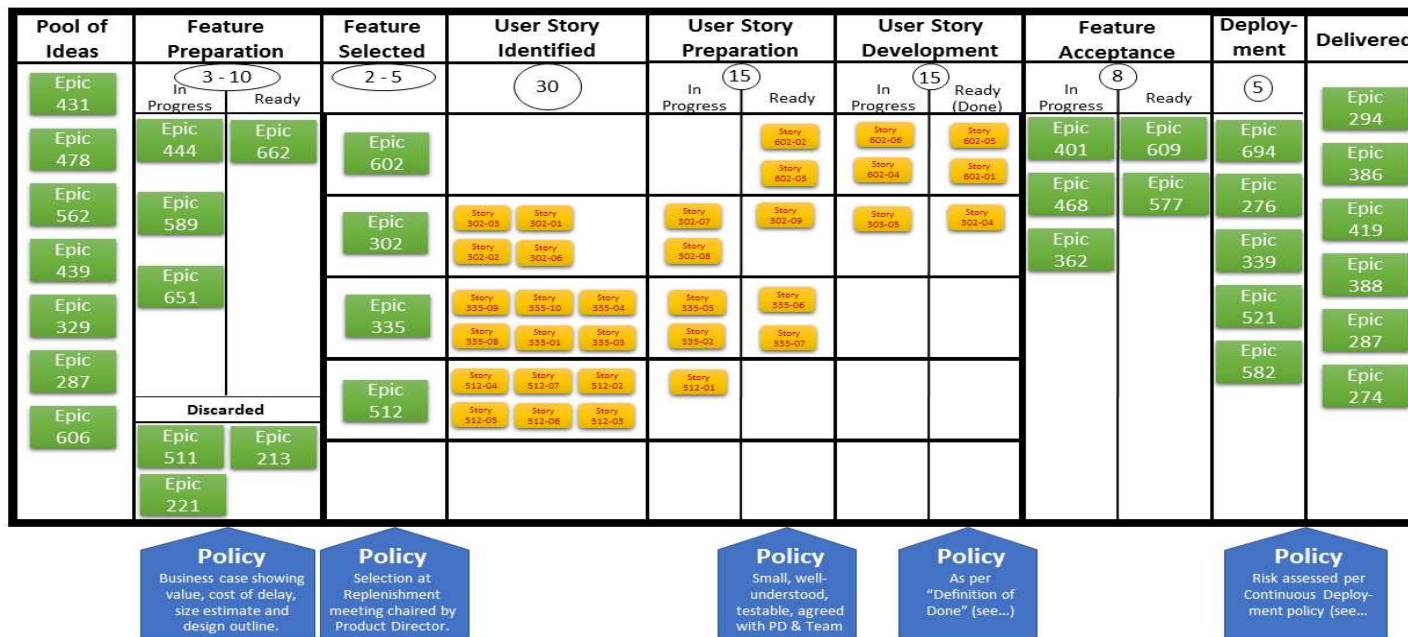
All the mentioned methodologies and philosophies are connected with each other and are the parts of Systems thinking

# Egy egyszerű Kanban Board



A popular example of a Kanban board for agile or lean software development consists of: Backlog, Ready, Coding, Testing, Approval, and Done columns.

# Kanban board a szoftverfejlesztési folyamathoz



By Andy Carmichael - Own work, CC BY-SA 4.0,  
<https://commons.wikimedia.org/w/index.php?curid=55448101>



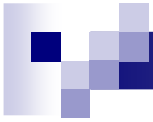
# Agilis és Kanban

- További , érdekes részletek itt:
  - <https://realtimeboard.com/blog/scrum-kanban-boards-differences/#.WXMdVFLeQO>



# Agilis és Lean: megéri!

- According to the Project Management Institute and its Pulse of the Profession 2015: Capturing the Value of Project Management 2015, 75% of highly agile organizations met their goals/business intent, 65% finished on time, and 67% finished within budget, which is higher than what organizations with low agility achieve. The same research shows that agile organizations grew revenue 37% faster and generated 30% higher profits than non-agile companies.
- Lean Startup principles have proven their effectiveness too. So thanks to lean approach, Dropbox went from 100,000 registered users to over 4,000,000 in 15 months; Wealthfront company now manages over \$200M and processes over \$2M a day; IMVU has reached 50 million registered users and a \$40+ million annualized revenue run rate.
  - [https://realtimeboard.com/blog/choose-between-agile-lean-scrum-kanban/#.WXL\\_vIFLeQM](https://realtimeboard.com/blog/choose-between-agile-lean-scrum-kanban/#.WXL_vIFLeQM)



# 3

## ALAPVETŐ IGAZSÁG

- KEZDET BEN NEM TUDNI ELŐRE MINDENT
- GARANTÁLT A VÁLTOZÁS
- MINDIG TÖBB FELADAT LESZ, MINT PÉNZ ÉS IDŐ



# NINCS

---

végző  
megoldás!



# Mit tehetünk, mégis?



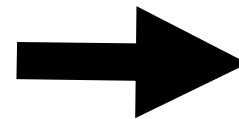
**LEAN**



**SCRUM**



**KANBAN**



# Mit tehetünk, mégis?



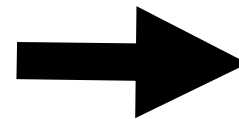
**LEAN**



**SCRUM**



**KANBAN**

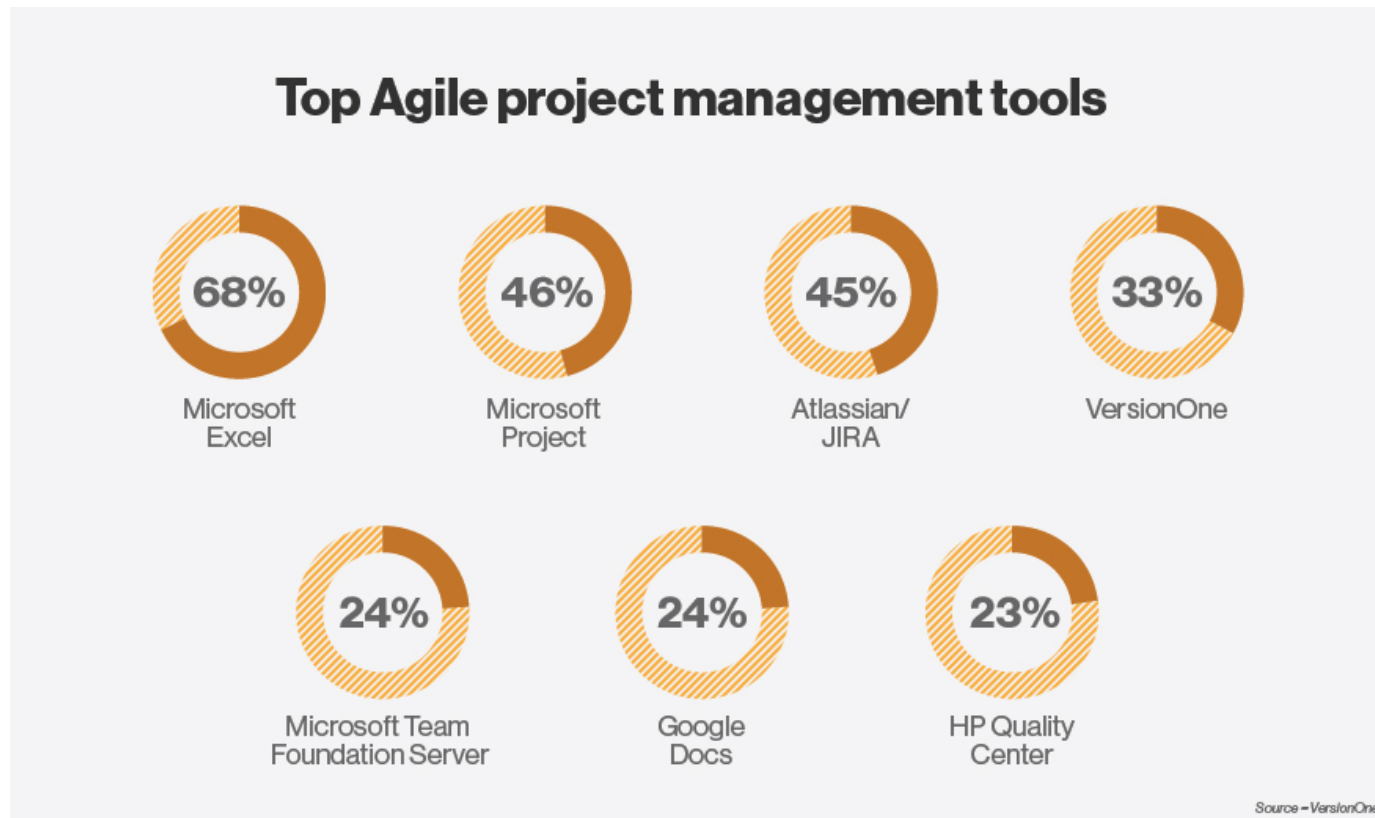


**+ saját  
megközelítés**





# Agilitást támogató eszközök



# Agilitást támogató eszközök

Scrum Board

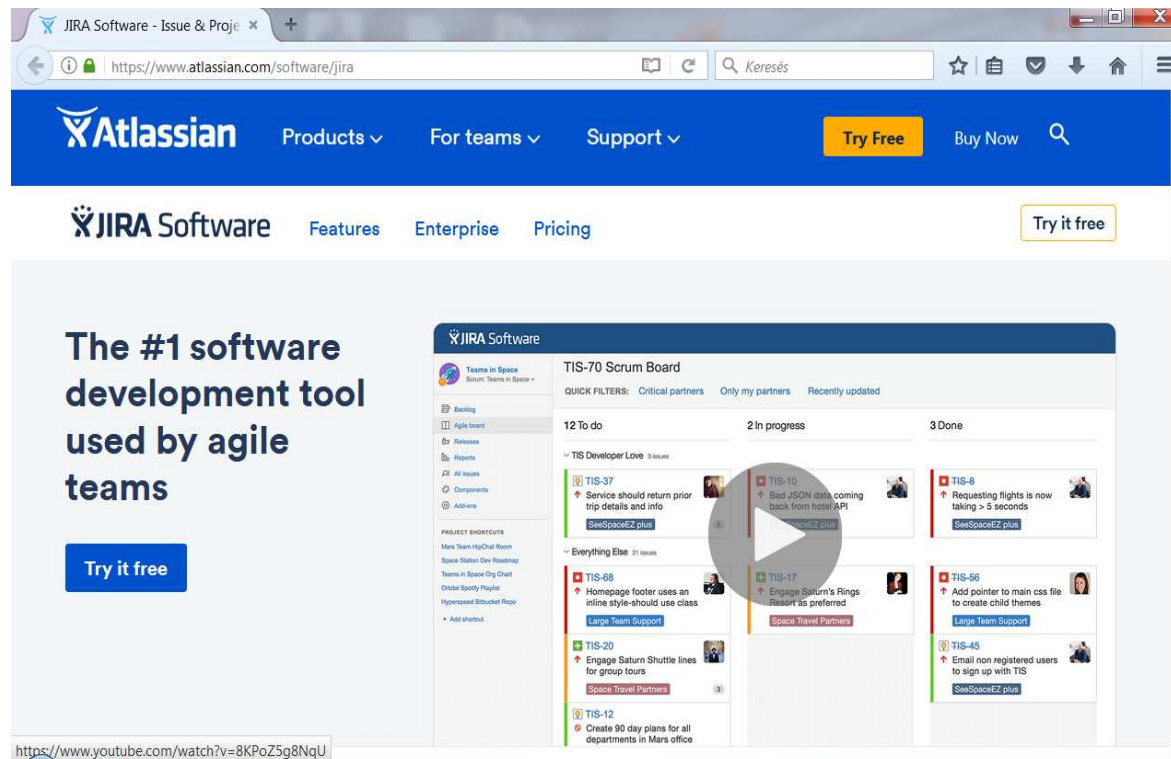
🕒 7 days remaining [Complete Sprint](#)

QUICK FILTERS: [Product](#) [UI](#) [Server](#) [Only My Issues](#) [Recently Updated](#)

2 To Do	3 In Progress	2 Code Review	1 Done
<div><div>🔴 TIS-68</div><div>Homepage footer uses an inline style - should use a class</div><div>Large Team Support</div><div>4</div></div>	<div><div>🟢 TIS-49</div><div>Draft network plan for Mars Office</div><div>Local Mars Office</div><div>5</div></div>	<div><div>🟢 TIS-69</div><div>Add a string anonymizer to TextUtils</div><div>Large Team Support</div><div>1</div></div>	<div><div>🔴 TIS-8</div><div>Requesting available flights is now taking &gt; 5 seconds</div><div>Large Team Support</div><div>2</div></div>
<div><div>🟢 TIS-16</div><div>Establish relationship with local office supplies company</div><div>Local Mars Office</div><div>9</div></div>	<div><div>🟢 TIS-17</div><div>Engage Saturn's Rings Resort as a preferred provider</div><div>Space Travel Partners</div><div>7</div></div>	<div><div>🔴 TIS-67</div><div>Developer Toolbox does not display by default</div><div>Large Team Support</div><div>5</div></div>	<div><div>🔴 TIS-56</div><div>Add pointer to main css file to instruct user to create child themes</div><div>Large Team Support</div><div>4</div></div>
	<div><div>🟢 TIS-30</div><div>Create Saturn Summer Sizzle Logo</div><div>Summer Saturn Sale</div><div>1</div></div>		<div><div>🟢 TIS-46</div><div>Email non registered users to sign up with Teams in Space</div><div>Summer Saturn Sale</div><div>8</div></div>
	<div><div>🟢 TIS-23</div><div>Engage JetShuttle SpaceWays for short distance space travel</div><div>Local Mars Office</div><div>1</div></div>		

# Agilitást támogató eszközök

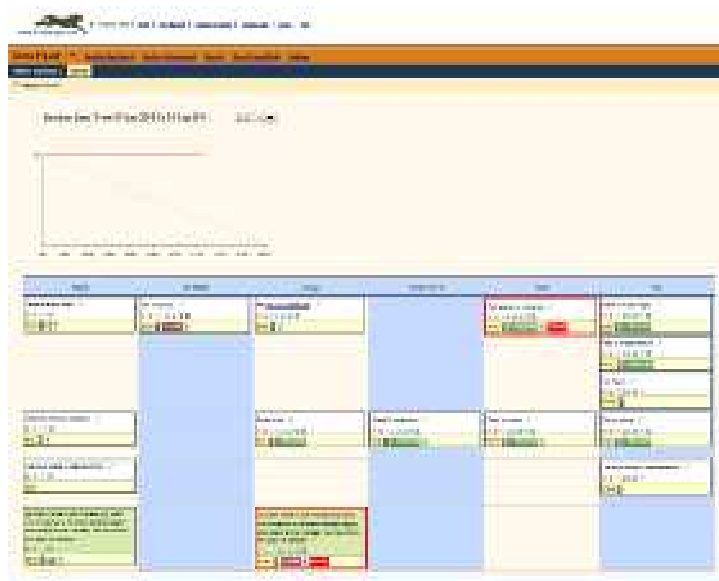
Atlassian



<https://www.atlassian.com/software/jira/agile>

# Agilitást támogató eszközök

## Silver Catalyst for Scrum



## Silver Catalyst for Kanban



<http://toolsforagile.com/> „A Lightweight, Enterprise Tool”



Konfliktusban van-e egymással az agilis és a hagyományos megközelítés?





# Miről volt szó...

- Szoftverfejlesztési / technikai / műszaki folyamatok
- Népszerű élelciklus modellek
  - vízésés, V-modell, spirál, iteratív, incrementális
- Hagyományos és agilis megközelítések
  - Agilis és Lean alapelvek
- Konfliktusban van-e egymással a hagyományos és az agilis szoftverfejlesztés?