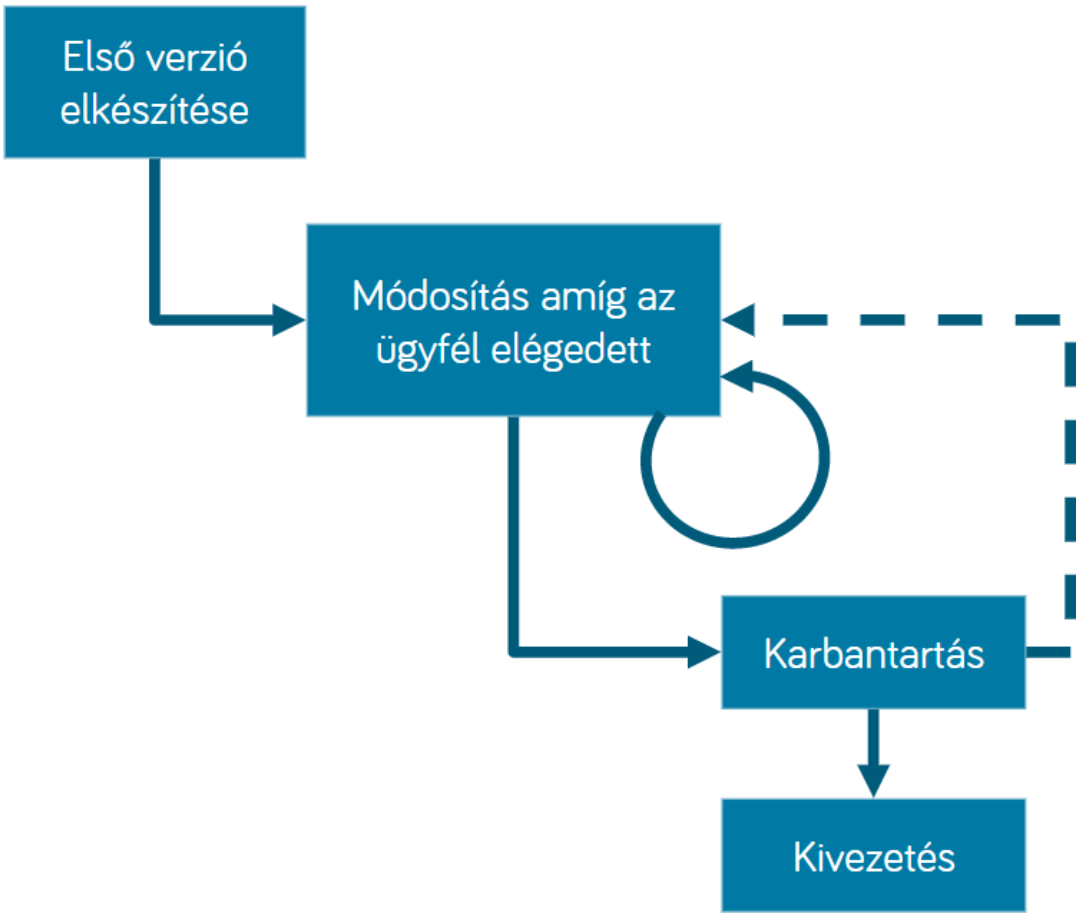
Szoftverfejlesztési folyamat

Modell

- **Tevékenységek strukturált halmaza**, amelyekkel létrehozunk egy szoftver rendszert: specifikálunk, tervezünk, kódolunk és tesztelünk.
- A szoftver folyamat modell a folyamat absztrakt leírása, ami segít eldönteni a résztvevőknek:
    - Milyen munkát csináljunk meg
    - Milyen sorrendben
    - A modell gondolkodni segít, nem merev előírások halmaza
    - Minden projekt saját egyedi folyamat mentén halad

Build and fix

- Nem kielégítő modell értelmes méretű projekt esetén.
- A költségek jelentősen megnőnek nagy projekt esetén.
- Tipikusan csúszik a szállítás.
- Nincs dokumentáció.
- A karbantartás nagyon nehéz specifikáció és tervek nélkül.
- Kis projektek esetén működik: kis menedzsment overhead.



A folyamat

- Sokfajta különböző folyamat, de mind tartalmazza az alábbi tevékenységeket:
  - Specifikáció: mit kell tudnia a rendszernek
  - Terv és implementáció: a rendszer elkészítése
  - Validáció: ezt akarja-e a megrendelő?
  - Evolúció: a megrendelői igények változásakor változtatjuk a rendszert
- A folyamat leírások kiterjedhetnek az alábbiakra is:
  - Termékek, amelyek egy folyamat eredményeként megszületnek
  - Szerepek, amelyek az emberek felelősségét tükrözik
  - Aktivitások elő-és utófeltételei

Tervezési megközelítések

Prototipizálás

- A prototípus a rendszer egy első verziója, ami koncepciók, dizájn alternatívák bemutatását célozza.
- **MVP:** Minimum Viable Product
- Segíti a követelmény felmérést és validációt.
- UI dizájn kialakításában az alternatívák felfedezésében.
- **Előnyei:**
  - Jobban használható szoftver
  - Közelebb áll ahhoz, amire a felhasználónak szüksége van
  - Jobb dizájn
  - Jobb karbantarthatóság
  - Kisebb fejlesztési költség
- A prototípusokat **el kell dobni** a valódi rendszer fejlesztése előtt:
  - Nem alakítható át úgy, hogy a nem-funkcionális követelményeknek megfeleljen.
  - A prototípusok általában **nem dokumentáltak**.
  - A prototípus felépítése nem megfelelő a sok változtatás hatására.
  - Általában nem felel meg az elvárt minőségi követelményeknek.

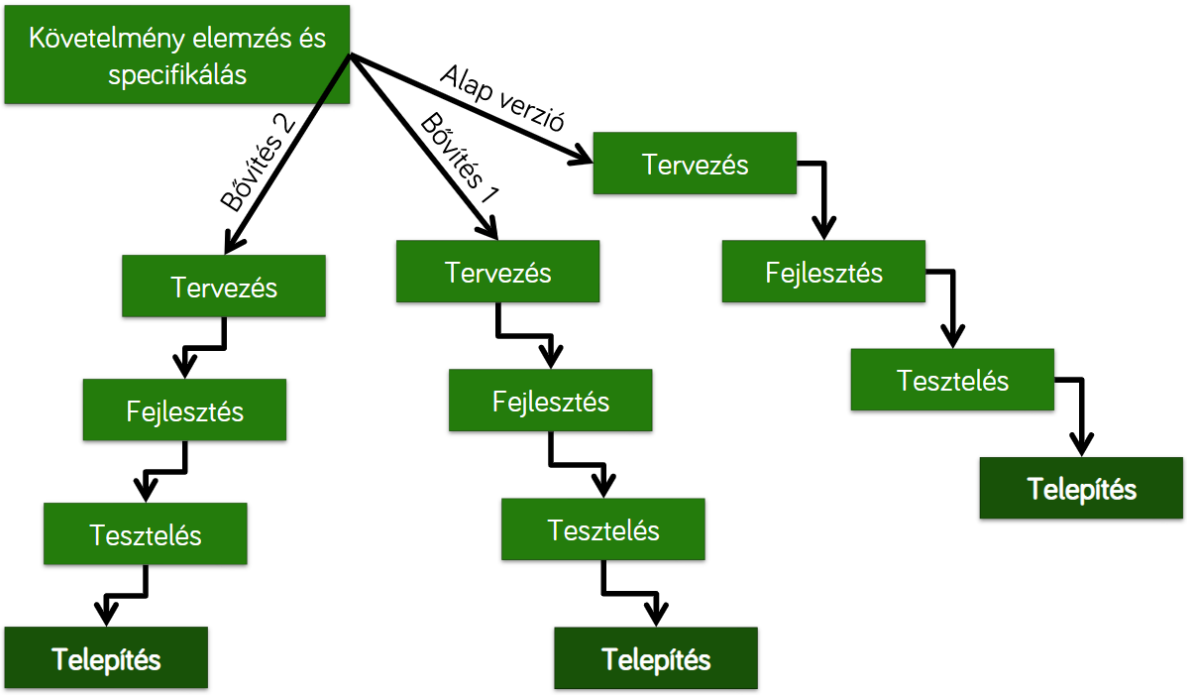
Vízesés modell

- **Terv-alapú modell**, különálló specifikációs és megvalósítási fázisok:
  - A víz az egyik teraszról folyik a másikra.
  - A terasz megtelik vízzel mielőtt a következőbe folyna.
  - A víz nem folyik felfelé amíg nem pumpáljuk.
  - A munka fázisok mentén folyik.
  - Mindegyik fázis lezárul mielőtt a következő elkezdődne.
  - Az előző fázisra nem térünk vissza, hacsak nem találtunk hibát a mostani fázisban.

- Az elmúlt 50 év leggyakrabban használt modellje, **ma is nagyon gyakori**, főleg hazánkban.
- **Lineáris modell:** egymást követik a lépések és csak akkor kezdünk el egy új fázist, ha az előző már befejeződött.
- **Iteratív vízesés modell:** van visszalépés az előző fázisra, ha hibát látunk.
- Megközelítése minden modellben megjelenik.
- Jól **rögzített határideje** van.
- Alfától-ómegáig minden ránk van bízva: mindent nekünk kell **kitalálni**, **megvalósítani**, **telepíteni**, **átadni** és **karbantartani**.
- A megrendelővel a projekt elején és végén találkozunk.
- **Lépései:**
    1. Követelmény specifikáció elkészítése
    2. Szoftvertervezés
    3. Megvalósítás és integráció
    4. Tesztelés és ellenőrzés
    5. Telepítés és karbantartás

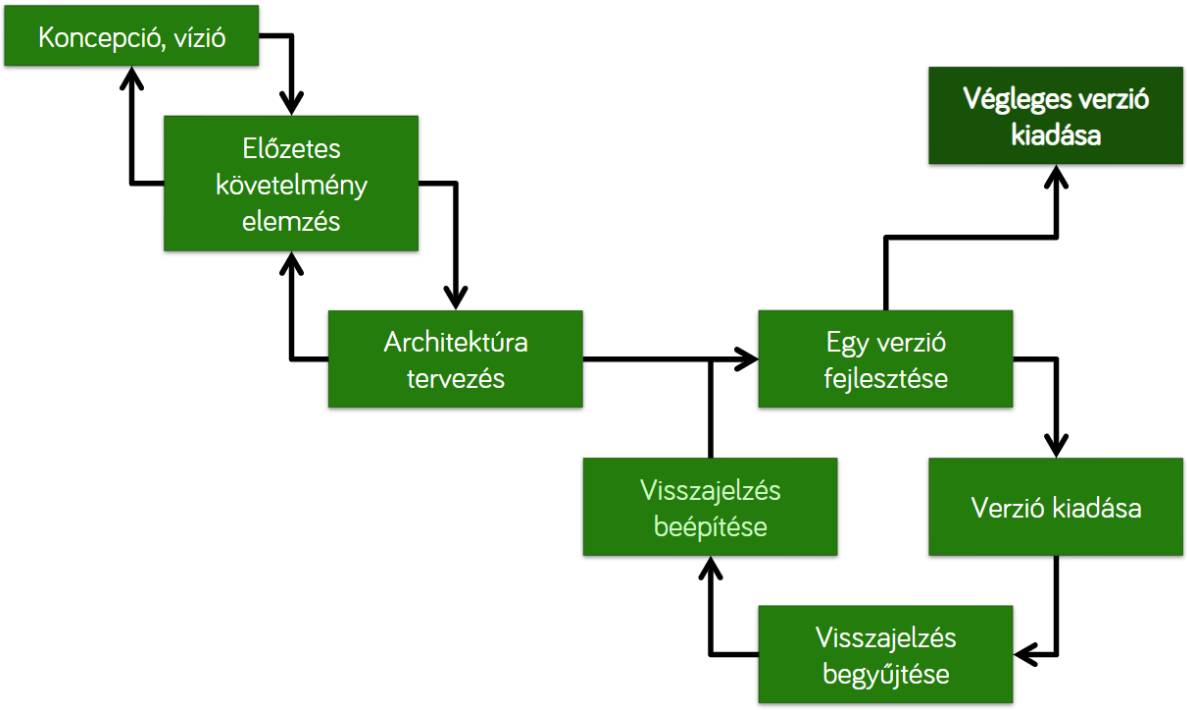
Inkrementális modell

- A specifikáció, fejlesztés és validáció átfedik egymást.
- Újrafelhasználás-központú módszer: a rendszert **meglévő komponensekből** építjük össze.
- Lehet **terv-alapú** vagy **agilis**.
- A gyakorlatban a legtöbb nagy rendszer mindegyik megközelítést használja.
- **Megismerjük a teljes követelményrendszert.**
- Ahelyett, hogy a rendszert egy lépésben szállítanánk le, azt **darabokra bontjuk** és inkrementumokban szállítjuk: az **inkrementumokat iteratív vízesés** modellben valósítjuk meg.
- A követelményeket **priorizáljuk**, a magas prioritásúakat szállítjuk elsőnek.
- Ahogy egy inkrementumot elkezdünk fejleszteni, a követelményeit már nem módosítjuk.
- **Inkrementális fejlesztés:**
    - Inkrementeket fejlesztünk és értékeljük azt mielőtt tovább lépnénk.
    - **Minimális változtatásokat eszközlünk.**
    - Az értékelést a **kulcs felhasználó**, üzleti elemző végzi.
- **Inkrementális telepítés:**
    - A **végfelhasználó** számára telepítjük az inkrementumot.
    - Valódibb visszajelzést kapunk a szoftverről.
    - Nehéz egy rendszer kiváltása esetén alkalmazni, mert eleinte kisebb a funkcionalitása mint a kiváltandó rendszernek.



Evolúciós modell

- **Induláskor nem ismert a teljes követelményrendszer.**
- Közös vízió és az architekturális alapok letétele után indul a folyamat.
- Mindig a **legfontosabb követelmények** kerülnek kidolgozásra, lefejlesztésre, tesztelésre és átadásra – ez egy **iteráció**.
- **Átadás után a visszajelzések** bekerülnek a követelménylistába.
- A szoftver minden iteráció után használható.



RUP (Rational Unified Process)

Jellemzők

- **A vízesés, inkrementális és újrahasznosító modellek ötvözése.**
- Iparban alaposan tesztelt legjobb gyakorlatok szoftverrendszer fejlesztéshez.
- **Adaptálható**nak tervezték: a projekthez igazítható.
- Átfogó folyamat-keretrendszer.
- Use case-vezérelt.
- Architektúra-központú.
- Javasolja az UML használatát.
- **Formálisabb** és előíróbb, mint más agilis keretrendszerek: jobban illeszkedik a klasszikus nagyvállalati kultúrába.
- **Kockázatok** folyamatos kezelése.
- Biztosítja, hogy **értéket** szállítunk az ügyfélnek.
- A **működő szoftverre** fókuszál.
- **Képes a változásokat korán befogadni.**
- Komponens alapon építkezik.
- **Csapatmunkában** gondolkozik.
- A **minőség** a folyamat része, nem utógondolat.

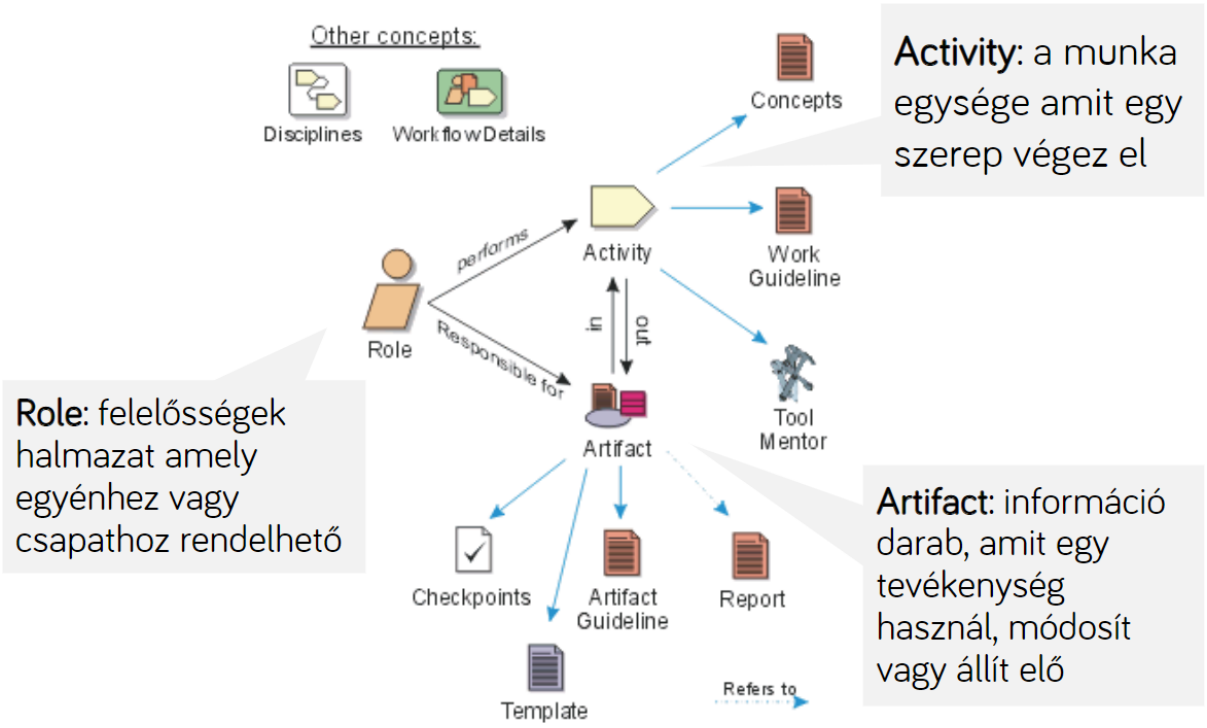
Fogalmak

- **Ciklus:** a szoftver egy kiadásához tartozó tevékenységek időbeli kerete.
- **Fázis:** minden ciklus négy fázisra osztott.
- **Iteráció:** a fázison belül több iteráció lehetséges, amely eredménytermékek kiadásával zárul. Az iterációk célja a kockázatok folyamatos csökkentése.
- **Mérföldkő:** egy iteráció vége, formális pont.
- **Workflow:** tevékenységek sorozata, amely látható értéket teremt.

Építőelemek

- **Szerepkörök (who):** a szerepkör képességet, kompetenciát és felelősséget definiál.
- **Eredménytermékek (what):** az eredménytermék (artifact) a feladatok eredményeit jelentik, beleértve a teljes folyamat során keletkezett modelleket és dokumentációt, például:
    - Vízió dokumentum
    - Prototípus
    - Use case
    - Üzleti leírás
    - Terv
    - Forráskód
    - Telepíthető állomány

- **Feladatok (how):** A feladat egy szerepkörhöz rendelt munkaegységet definiál, melynek használható eredménye van.
- Minden iteráción belül a feladatok 9 területre oszlanak fel:
    - 6 **mérnöki** területre:
        - Üzleti modellezés
        - Követelmények
        - Elemzés és tervezés
        - Implementáció
        - Tesztelés
        - Üzembe helyezés
    - 3 **támogató** területre:
        - Konfiguráció és változáskövetés
        - Projektmenedzsment
        - Környezet



Életciklus fázisai

- Az iteratív fejlesztési folyamatot **4 fázisra** osztja.
- Minden fázis **egy vagy több iterációt** tartalmaz.
- Minden fázis alatt minden terület megjelenik, de más mértékben.
- Tipikusan ciklusonként kötünk szerződést, egy ciklus néhány hónap hosszú.
- Egy iteráció tipikusan **2-6 hét** hosszú.
- A bővített modell tartalmaz plusz egy fázist: üzemeltetést és a hozzá tartozó folyamatot.

- **Inception (kezdet):**
  - A projekt terjedelem kialakítása – milyen hozzáadott értéke van a felhasználó szempontjából, illetve mennyi feature-t valósítunk meg.
  - A csapat eldönti, hogy érdemes-e a projektet elindítani, illetve folytatni.
  - Elindul a tervezés:
    - Költség elemzés, erőforrástervek
    - Kockázatok
    - Folyamatok, eszközpark kialakítása
    - Magas szintű architektúra javaslat
- **Elaboration (kidolgozás):**
  - A probléma terület megértése
  - Prototípusok készítése
  - Funkcionális és nem-funkcionális követelmények rögzítése, finomítása:
    - Use case-ek és kiegészítő dokumentáció
    - Szekvencia és kollaborációs diagrammok
  - A projekt architektúra pontosítása
  - Részletes projekt terv kialakítása (erőforrás igények, függőségek, stb.)
  - Kockázatok felülvizsgálata
- **Construction (megépítés):**
  - Működő szoftver készítése
  - A hagyományos fázisok közül itt történik:
    - A szoftver tervezése – további UML és egyéb mérnöki tervek, kevesebb szöveges dokumentáció
    - A szoftver implementációja és fejlesztése
    - A szoftver tesztelése
- **Transition (átadás):**
  - A szoftver publikus kiadása és átadása:
    - Alfa, béta tesztelés, pilot időszak
    - Végső simítások
    - Visszajelzések alapján történő korrekciók
  - Dokumentáció véglegesítése
  - A fejlesztő csapat mérete csökken
  - Az irányítást átadjuk az üzemeltetésnek

6 best practices

1. **Develop iteratively (iteratív fejlesztés)**
    - Megrendelői prioritás alapján tervezzük a következő iterációt.
    - Minden fázis végén go/no go döntés: költségcsökkentés.
2. **Manage requirements (követelmény kezelés)**
    - Mindig tartsuk szem előtt, hogy a követelményeket a felhasználók határozzák meg.
    - A követelményeket és változásokat dokumentáljuk.
3. **Use components (komponensek használata):** A projekt komponensekre bontása segíti a tesztelést, újra felhasználást és átlátást.
4. **Model visually (vizuális modellezés):** Használjunk diagramokat a szoftver statikus és dinamikus nézetének leírására.
5. **Verify quality (minőségtudatosság)**
    - Biztosítsuk, hogy a szoftver a szervezet minőségi követelményeinek megfelel.
    - A tesztelés mindig kapjon kitüntetett szerepet a projekt minden fázisában.
    - A tesztelési feladat folyamatosan nő a projekt előrehaladtával, de konstans faktornak kell lennie!
6. **Control changes (változások követése és ellenőrzése)**
    - Kövessük és dokumentáljuk a változásokat, használjunk erre eszközöket.
    - Számos projektet több különböző csapat fejleszt, gyakran különböző helyszínen, más-más platformok használatával.
    - Folyamatosan gondoskodjunk a változások szinkronizálásáról és verifikálásáról: Continuous Integration (CI).