Követelmények

Megértése

- **Több résztvevő**
  - **Fejlesztő:** fejlesztéshez ért, az üzleti területhez nem.
  - **Megrendelő:** üzletileg érdekelt a szoftverben, de nem feltétlen ért se a szoftverhez, se az üzleti területhez.
  - **Domain expert, felhasználó:** értene a doménhez, talán tudja, mi segítené a munkáját, de nem mindig kérdezik meg elégszer, illetve nem is feltétlen tudja elmondani elsőre.
  - **Megrendelő ≠ felhasználó:** sokszor nem azonosak, pl. kórházigazgató és ápolónő.
- **Rossz irányba ment fejlesztés**
  - **Frusztráció** a fejlesztőnek
  - **Pazarlás** és időbeli csúszás a megrendelőnek
  - **Üzleti nehézség** vagy bukás
- **Iteratív fejlesztés**
  - Gyakori validáció: „ez az, amit szerettél volna?”
  - Nem kerüli el a félreértéseket, de kordában tarthatóak.
  - Más megközelítést igényel: nem lehet rétegenként haladni, horizontálisan kell felosztani a feladatot.

Változása

- Még ha meg is értjük egymást, sok minden változhat:
  - Jogi környezet
  - Piaci verseny helyzete
  - Technológiai fejlődés
- A megrendelőnek kell tudnia, hogy versenyképes-e, amit szeretne.
- Nekünk felkészülni az esetleges változásokra: lehet holnap már mást kér tőlünk.
- Erre is az iteratív fejlesztés a megoldás.

Verifikáció és validáció

- **Verifikáció:** a specifikációval összhangban implementálom-e a rendszert?
  - **Jól építem-e meg?**
  - Klasszikusan ezt teszteljük.
- **Validáció:** az üzleti igényeket kielégítő rendszert építek-e?
  - **A jó rendszert építem meg?**
  - Jól lettek definiálva a követelmények?
  - Felhasználó előtti demóval vagy speciális tesztekkel végezzük.

Priorizálás

- **Problémák:**
  - Tapasztalat kell a becsléshez.
  - Lehetnek nem várt kihívások, érdemes ráhagyással tervezni.
  - Elnagyolt, változó vagy félreértett követelmények tovább nehezítik a helyzetet.
  - Nehezen tartható keretek.
  - Ha kifutunk az időből, akkor nem biztos, hogy továbbra is lesznek anyagi erőforrások a fejlesztés fedezésére.
  - Csúszással nem csak nőnek a közvetlen költségek, hanem a korábbi piacra lépésből realizálható haszontól is elesik
- A fenti problémák **priorizálással mérsékelhetők**.
- Mindig a **legfontosabb elemeken dolgozunk**: ami elkészül, az legyen jó és használható.
- Ezek minél előbb elkészülnek – ha valamire végül nem jut pénz vagy idő, az kisebb jelentőségű dolog.
- Kevésbé kerül veszélybe a piacra lépés.
- **Hogyan?**
  - Azt teljesítsük, amit kérnek, ne próbáljuk túlteljesíteni.
  - Gondolkodjunk előre, de ne túlzottan.
  - Inkább hagyjuk nyitva a bővítés lehetőségét, de ne fektessünk sok energiát túl előre a tervezésbe.

Agilis módszertanok

Áttekintés

- Figyelembe veszik a fenti nehézségeket.
- **Csapatmunka**, mindenki tud mindenről.
- A **felhasználót bevonva** dolgozunk: látható számára, hol tartunk ⇒ **bizalom**.
- Minimális tervezéssel induláskor.
- **Változásra készen**, azokat befogadva: rendszeresen vizsgáljuk meg, hogy jó-e az irány.
- Gyors fejlesztés – **gyakori kiadások**:
  - **Iteratívan**, mindig a legnagyobb prioritásokra fókuszálunk.
  - Az instabil követelmények nem tartják fel a fejlesztést.
- Pehelysúlyú, de erősen keretezett módszertanok.
- **Egy cél:** minden fejlesztő ugyanúgy lássa a rendszert, ahogy a majdani felhasználók is látni fogják.

Irányelvek

1. **Egyének és interakció** inkább, mint *folyamatok és eszközök*.
2. **Működő szoftver** inkább, mint *jól dokumentált szoftver*.
3. **Felhasználó bevonása** inkább, mint *szerződések tárgyalása*.
4. **Reagálás a változásokra** inkább, mint *egy terv követése*.

Scrum

Áttekintés

- Egy konkrét **agilis keretrendszer**.
- Előír **szerepköröket** és **eseményeket**, de további technikák is belevehetők, amelyek nem mondanak ellent az alapvetéseinek.
- **Időben keretezett** eseményekkel dolgozik, ami alatt be kell fejezni őket.

A csapat

- Egy csapat, közös felelősség, három szerepkör.
- **Product Owner (PO):** a megrendelőt képviseli.
    - Átmenet az üzleti és szoftveres világ közt.
    - Érti a követelményeket és **közvetíti** a fejlesztőknek.
    - **A csapat része.**
- **Scrum Master (SM):** a Scrum-elvek betartását segíti.
    - Nem klasszikus PM.
    - Nem a csapat vezetője.
    - A Scrum-ban képzett.
- **Development Team (DT):** a fejlesztők szigorú szerepkörök nélkül.
- Önmenedzselő csapat:
    - A **Development Team** önmenedzselő.
    - Ők tudják a legjobban, hogyan tudnak hatékonyan dolgozni.
    - Ki, hogyan, min dolgozik.
    - Minden taszkért közösen felelnek.
    - A **Scrum Master** és **Product Owner** része a csapatnak.

Események és megbeszélések

Sprint Planning + Sprint

- **Párórás meeting** a sprint elején.
- A sprint tipikusan **2-4 hét**, erre tervezünk, ideális napokkal és hetekkel (napi 8 óra, heti 5 nap)
    1. A user storykon megyünk prioritási sorban.
    2. Beazonosítjuk az alfeladatokat.
    3. Hozzárendelünk egy story pontot(Adunk egy időbecslést).
    4. Amikor elég story-t „beáraztunk”, vállalást teszünk.

Daily Scrum

- Napi szinten is tervezünk a nap elején.
- Mindenki képben van a teljes státusszal, a felmerülő problémákat rövidtávú tervezéssel lehet kezelni.
- Max. **15 perc**, ténylegesen felállva.
- Mindenki az alábbi kérdésekre válaszol:
    - Mit végeztem el tegnap?
    - Mit fogok ma csinálni?
    - Van-e valami, ami akadályozza a munkámat?

Sprint Review (Demo)

- Annak áttekintése, hogy mely munkák készültek el és melyek nem.
- Az elkészült munka bemutatása a terméktulajdonos és a fejlesztésben érdekeltek részére (demo).

Sprint Retrospective

- Cél a munkafolyamataink folyamatos javítása:
    - Mit csináltunk jól, mit kell megtartani?
    - Mit kéne másképp próbálnunk?
    - Nem bűnbakok kereséséről szól.
- **Kimenet: végrehajtható akciók**
    - Legalább egyet fel kell venni a következő sprint backlogjába.
    - Teszteljünk többet (pl. csak getter, setter és konstruktor maradhat teszteletlen)