Emich Szabolcs

Agilitás

Cynefin

Komplex

Kísérlet

Megfigyelés

Válasz

Kibontakozó

Komplikált

Megfigyelés

Analizálás

Válasz

Jó gyakorlat

Kaotikus

Cselekvés

Megfigyelés

Válasz

Új

Egyértelmű

Megfigyelés

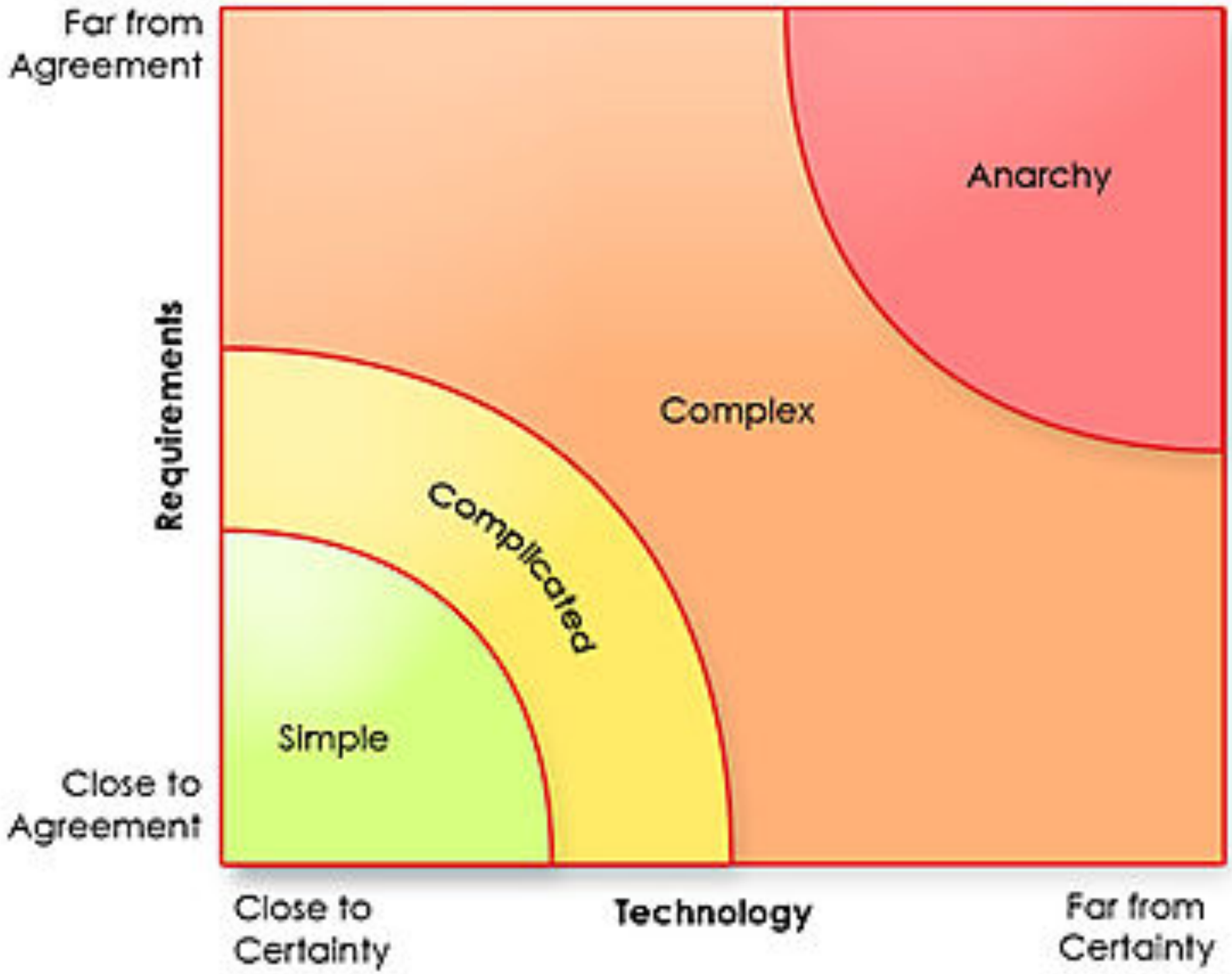
Besorolás

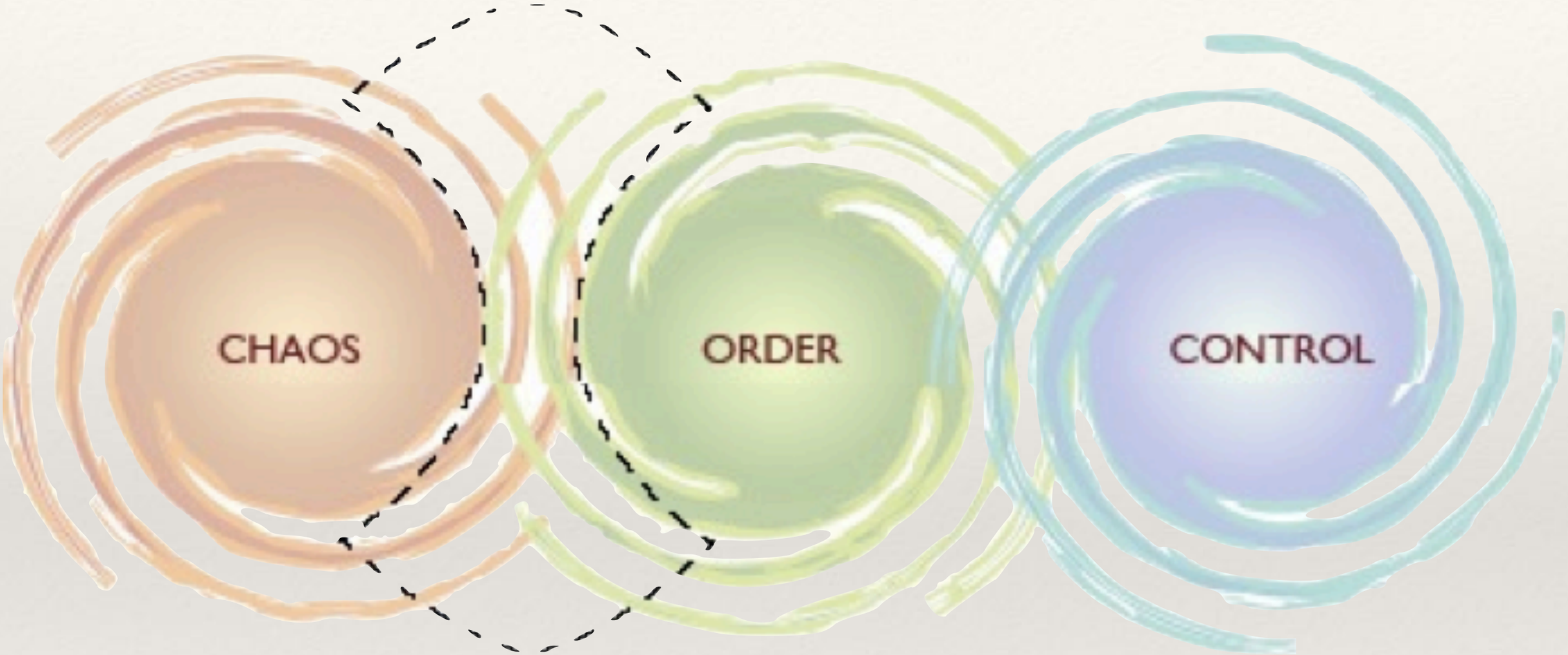
Válasz

Legjobb módszer

Emich Szabolcs - sz@bolcs.hu

The Spectrum of Process Complexity

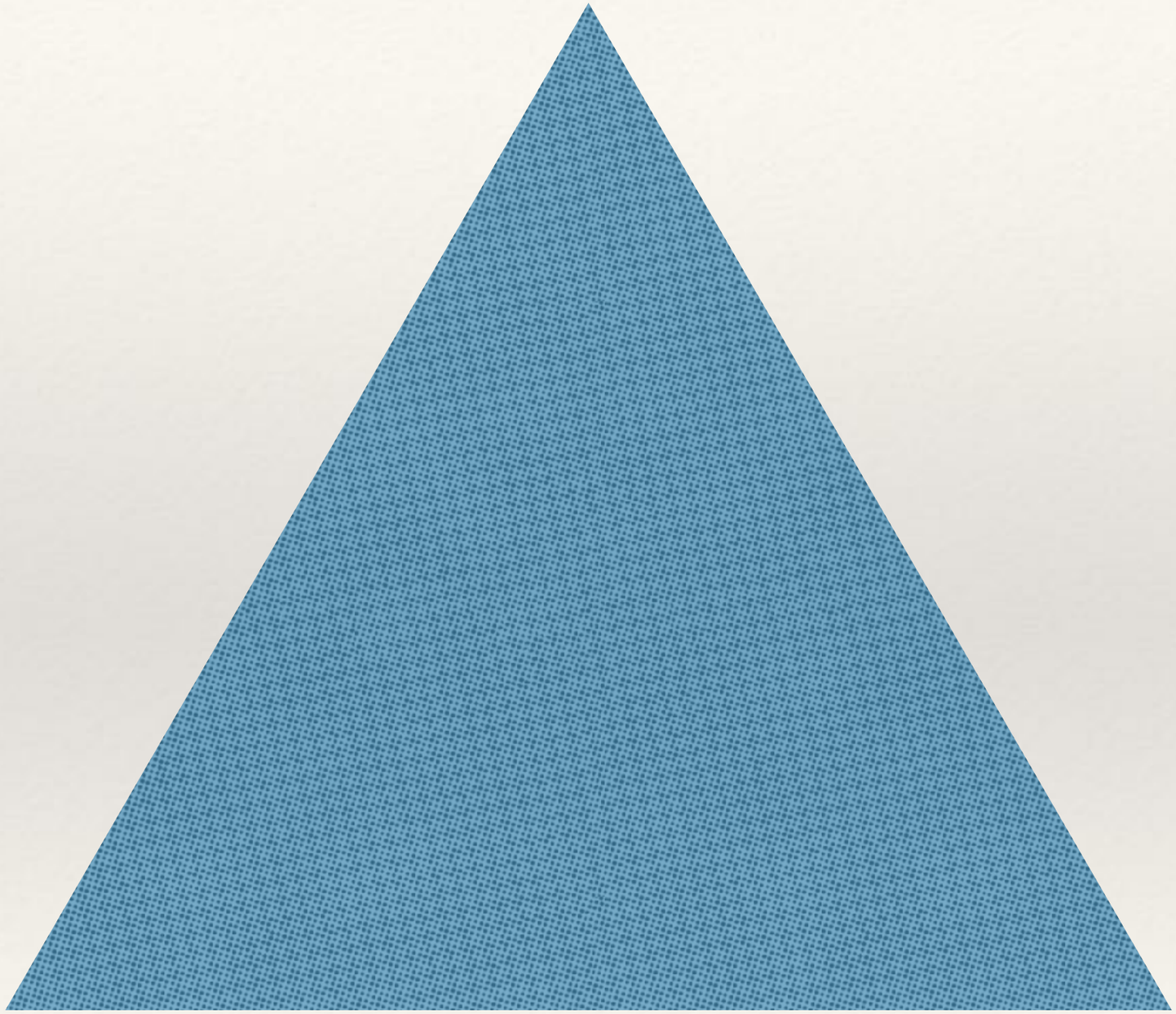




"The Chaordic Field"

Emich Szabolcs - sz@bolcs.hu

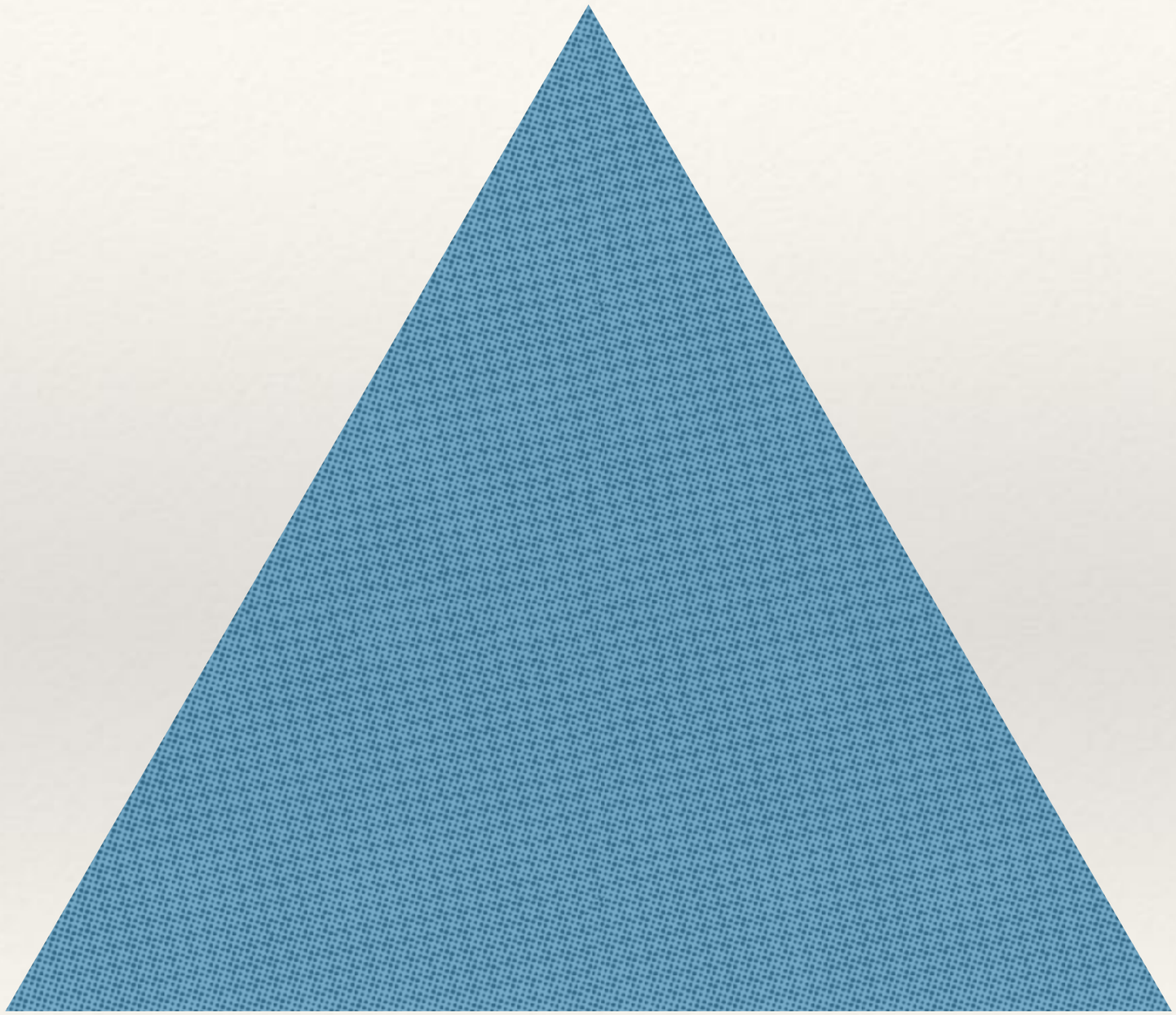
A bűvös háromszög



Emich Szabolcs - sz@bolcs.hu

A bűvös háromszög

Minőség

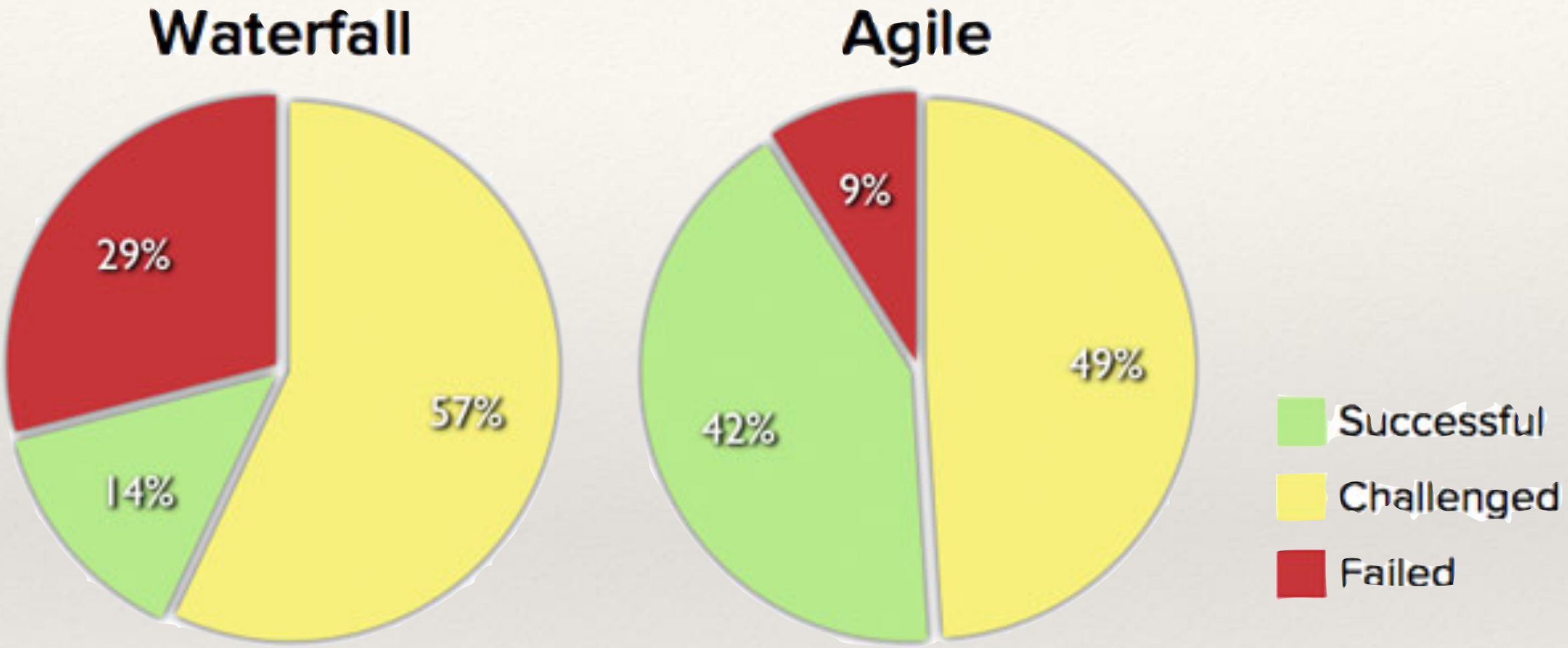


Funkciók

Idő

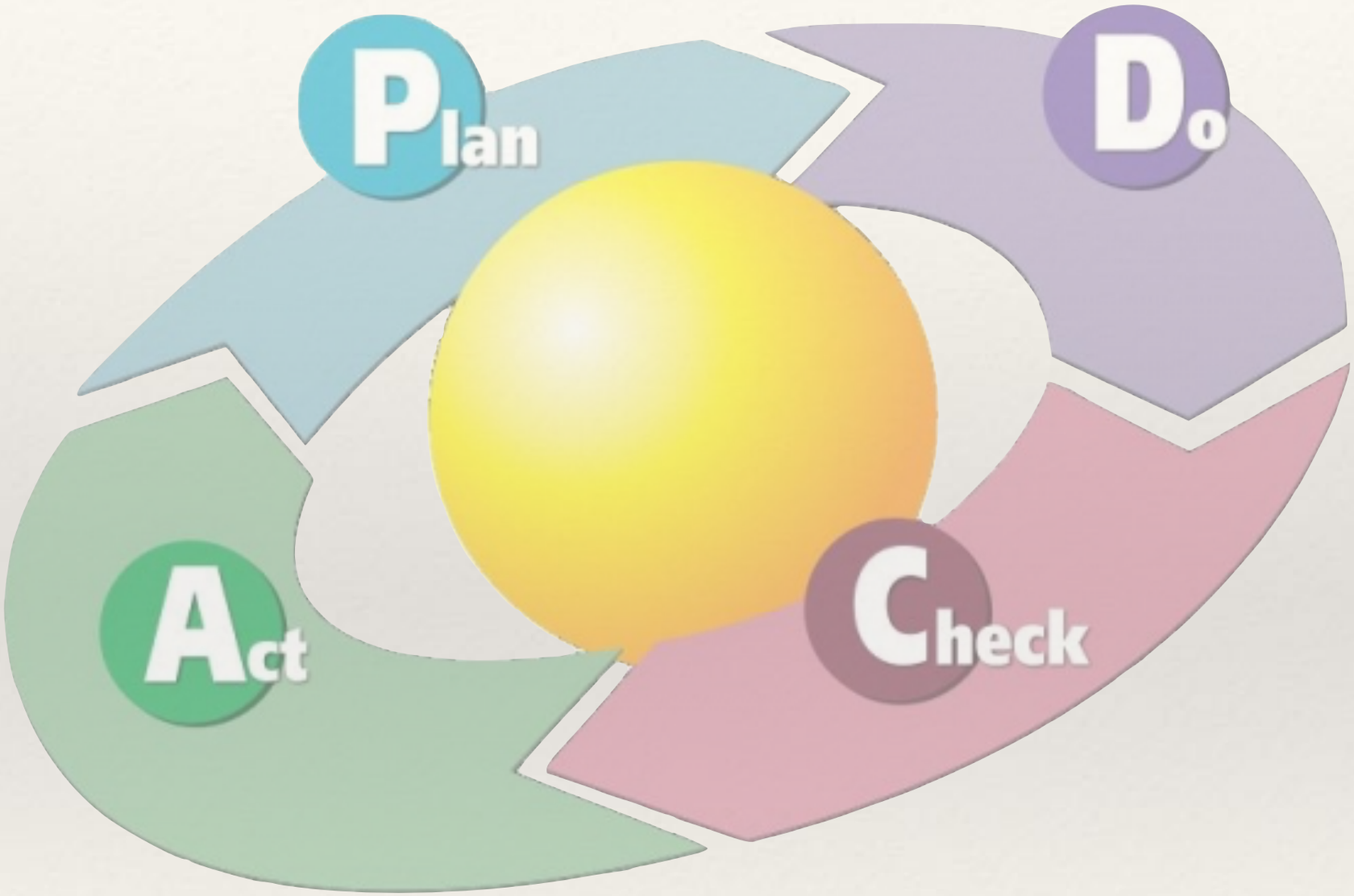
Emich Szabolcs - sz@bolcs.hu

Miért?



Source: The CHAOS Manifesto, The Standish Group, 2012

PDCA ciklus



Emich Szabolcs - sz@bolcs.hu

Agilis folyamat

- Egy empirikus folyamat, ahol a környezet és a követelmények folyamatosan változnak
- Nincs legjobb módszer, a rendszer komplex
- A siker alapja:
  - gyakori és korai visszajelzések az elkészült termékről
  - a résztvevők alkalmazkodhassanak a visszajelzésekre
  - hatékony kommunikáció a résztvevők között

Empirikus folyamat irányítási környezet

- A komplex promlémák megoldása alkalmazkodást igényel, ehhez a következők kellenek:
  - kreativitás
  - kezdeményező készség
  - tanulni képes egyének és csapatok
- Szükség van olyan kultúrára, ahol alapértékek a:
  - bizalom
  - megbecsülés (az emberek nem erőforrások)

Miért vezet be valaki Agilitást?

- kiszámíthatatlan és folyamatosan késő munka
- folyamatosan változó, félreérthető prioritások
- hatékonytalan meetingek
- minőségi problémák, sok hiba
- alacsony morál és elkötelezettség, “nem az én dolgom” hozzáállás
- silósodás, bizalmatlanság, kommunikációs problémák
- lassú termékfejlesztés

Agilis kiáltvány (2001)

Mi a jobb szoftverfejlesztés módjait fedezzük fel azáltal, hogy fejlesztünk és segítünk másokat fejleszteni. E munka során értékesebbnek tartjuk:

- **Az egyént és a személyes kommunikációt** a módszertanoknál és az eszközöknél.
- **A működő szoftvert**, az átfogó dokumentációnál.
- **A megrendelővel való együttműködést**, a szerződéshez való merev ragaszkodással szemben.
- **A változásra való reagálást**, a tervek rigorózus követésével szemben.

Noha, fontosak az utóbbiak is, mi fontosabbnak tartjuk az előzőeket.

Az Agilitás 12 alapelve

- 1) Legfontosabb a vevőnk elégedettsége, ezért értékes terméket szállítunk neki minél előbb és folyamatosan.
- 2) Elfogadjuk, hogy a követelmények változhatnak akár a munka vége felé is. Az agilis módszertanok befogadják a változást a megrendelő versenyképességének érdekében.
- 3) Gyakran szállítunk működő terméket, pár hetes vagy hónapos időközönként, lehetőleg a rövidebb periódust választva.
- 4) A megrendelők, üzleti szakemberek és a megoldás szállítók minden nap együtt dolgoznak a teljes projekt során.

Az Agilitás 12 alapelve

- 5) Motivált csapattagokkal dolgozunk. Létrehozzuk számukra a szükséges környezetet és támogatást, és bízunk bennük, hogy elvégzik a munkát.
- 6) A csapaton belül a személyes beszélgetés az információ átadásának leghatásosabb és leghatékonyabb módja.
- 7) Az előrehaladás elsődleges mércéje a működő termék.
- 8) Az agilis módszertanok elősegítik a fenntartható fejlesztést. A szponzoroknak, kivitelezőknek, felhasználóknak korlátlan ideig képesnek kell lenniük az egyenletes sebesség megőrzésére.

Az Agilitás 12 alapelve

- 9) A technikai kiválóságra és a jó tervezésre fordított figyelem 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.

Agilis téveszmék

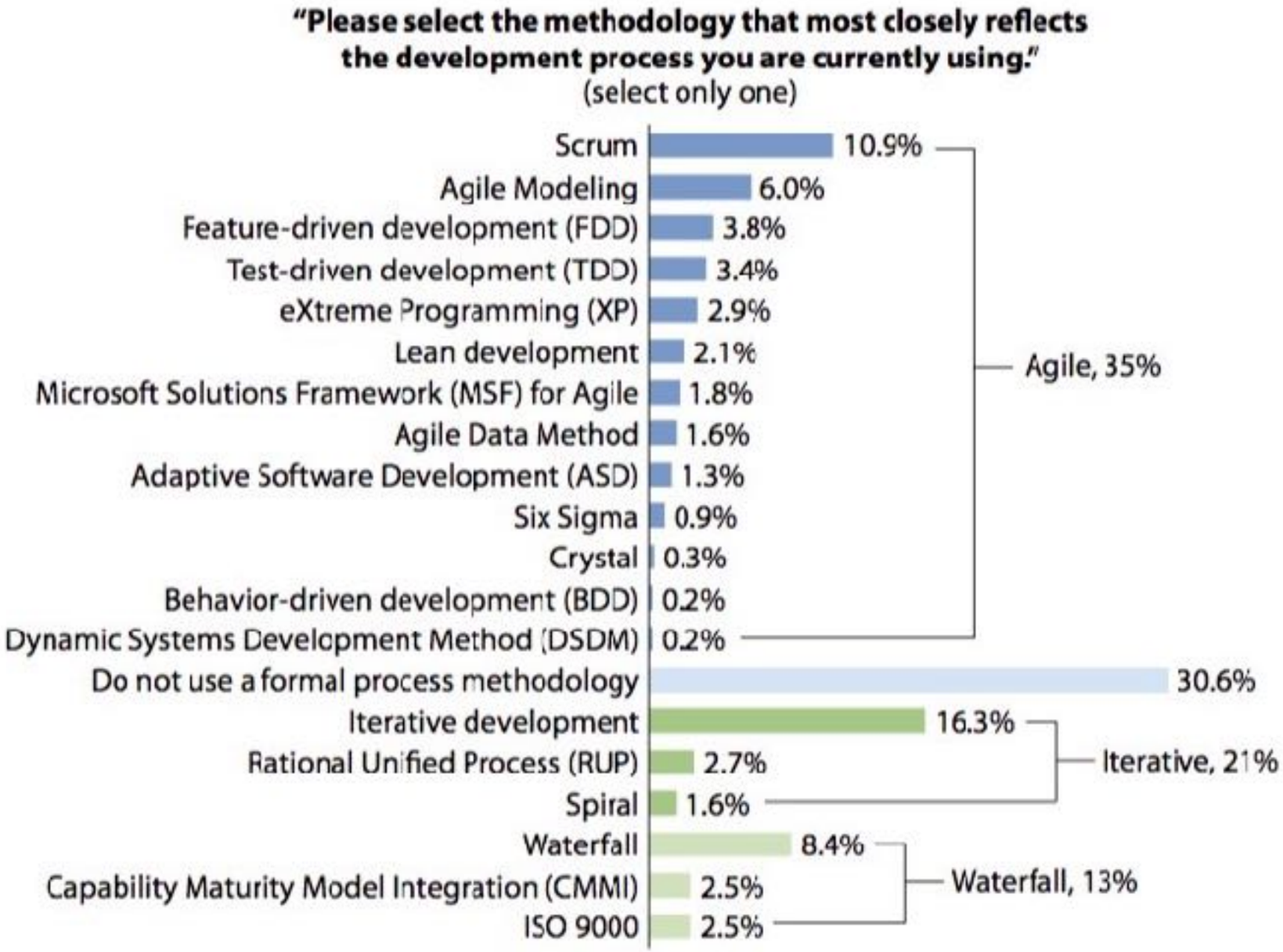
1. **Az agilis működésben nincs dokumentáció**
   - Annyi dokumentáció van, amennyi szükséges. A dokumentáció is a leszállított megoldás része. Ha értéket hordoz, készítsd el!
2. **Az agilitás valami új dolog**
   - Nem éppen. Az Agilis kiáltvány 2001 óta létezik. A Scrum alapjairól már 1998 -ban írtak, és számos gyakorlat az XP -ből 1995 -ben már létezett
3. **Az agilitásban nem kell tervezni**
   - Dehogynem. Sőt, többet kell tervezni! A tervezés folyamatos a munka során. Az viszont igaz, hogy az agilitás nem javasolja a távoli jövőben előkerülő funkciók tervezését, amik a munka megkezdése után már nem aktuálisak.

Agilis téveszmék

4. **Az agilis csapatokban az emberek azt csinálnak amit csak akarnak**
   - Az agilis munka jóval nagyobb fegyelmet igényel, mint a korábbi módszertanok.
5. **A SCRUM és a KANBAN egymás ellenfelei**
   - Ez nem igaz, sőt agilis csapatok gyakran alkalmaznak elveket mindkettőből
6. **Az agilitás csak új termék fejlesztésénél előnyös**
   - Az agilitás minden gondolkodást igénylő projekt esetén előnyös. Az agilitás lehetővé teszi, hogy egyre több értéket állítsunk elő egyre kevesebb idő alatt

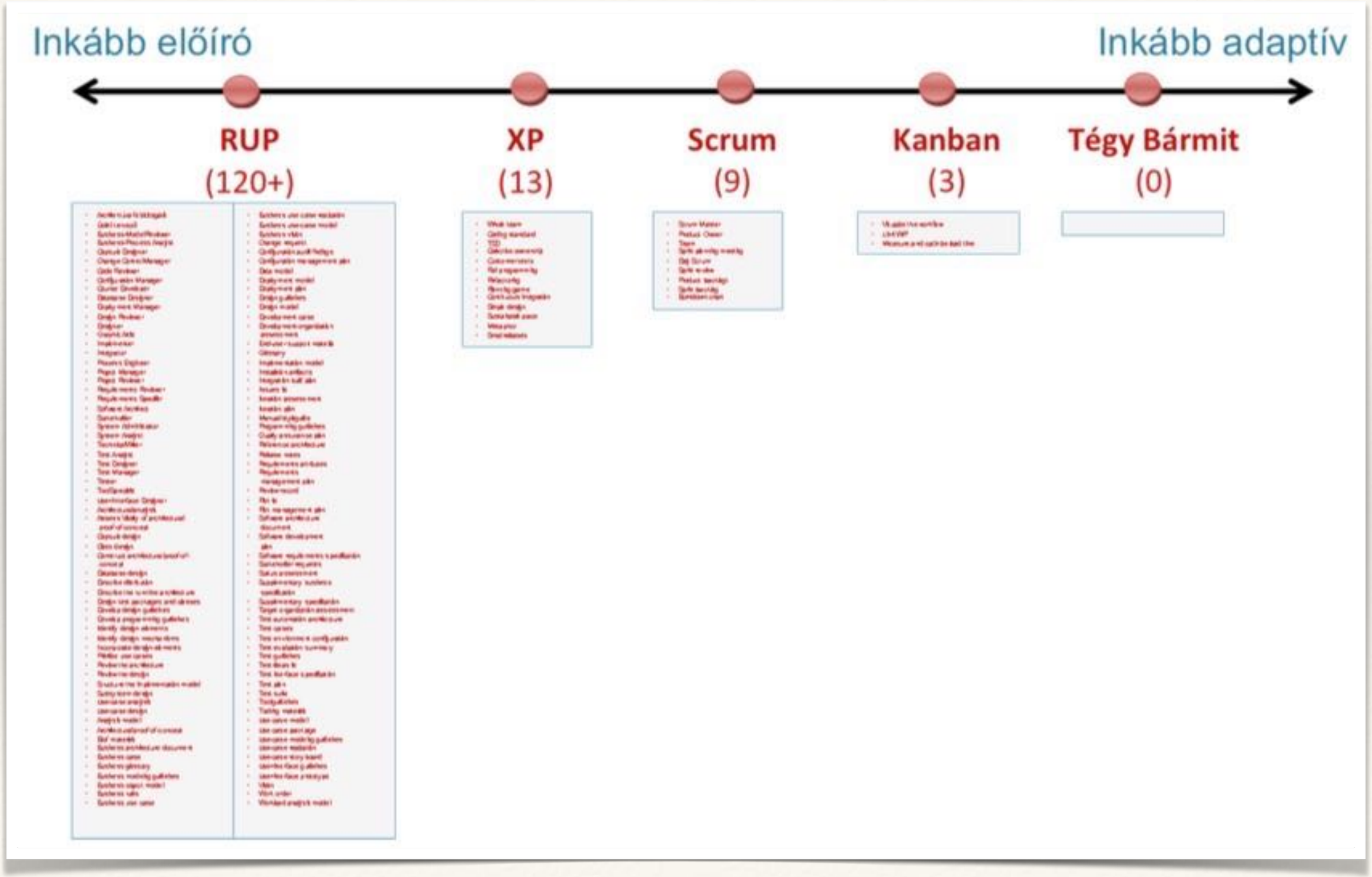
Hol használják?

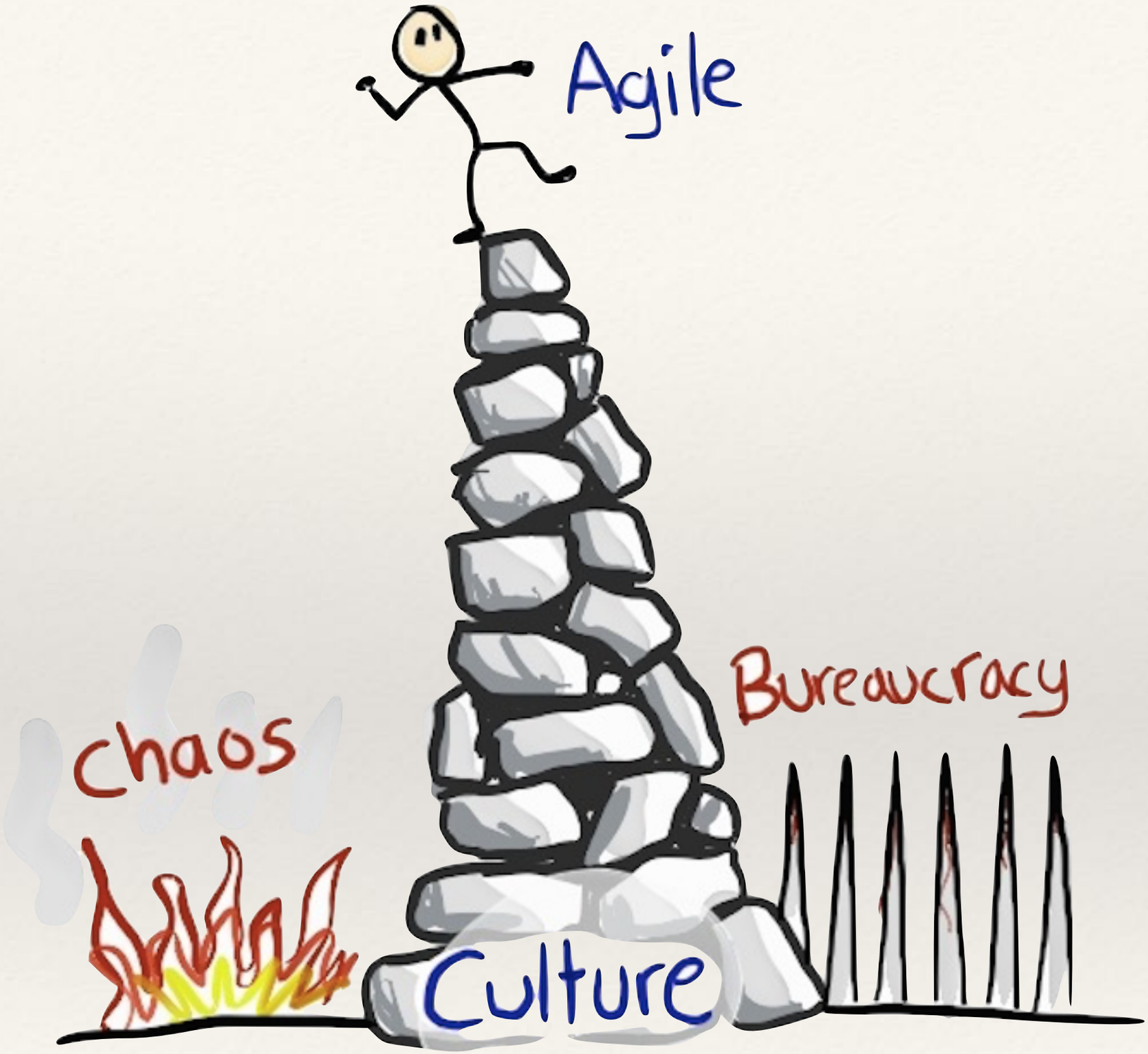
Figure 1 Agile Is Organizations' Primary Development Approach



56100 Source: Forrester Research, Inc.

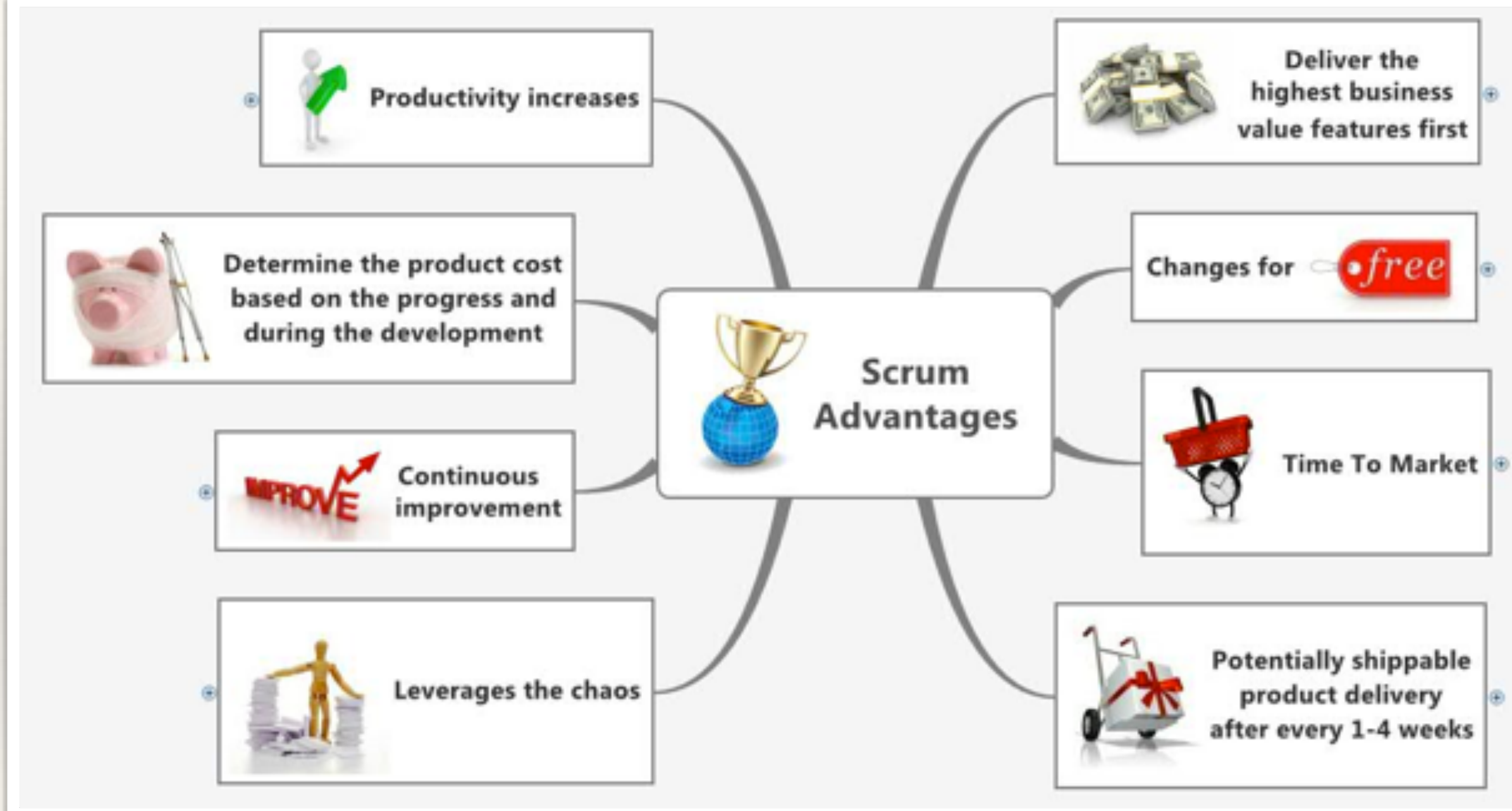
Mennyire bonyolult?




Agile
chaos
Bureaucracy
Culture

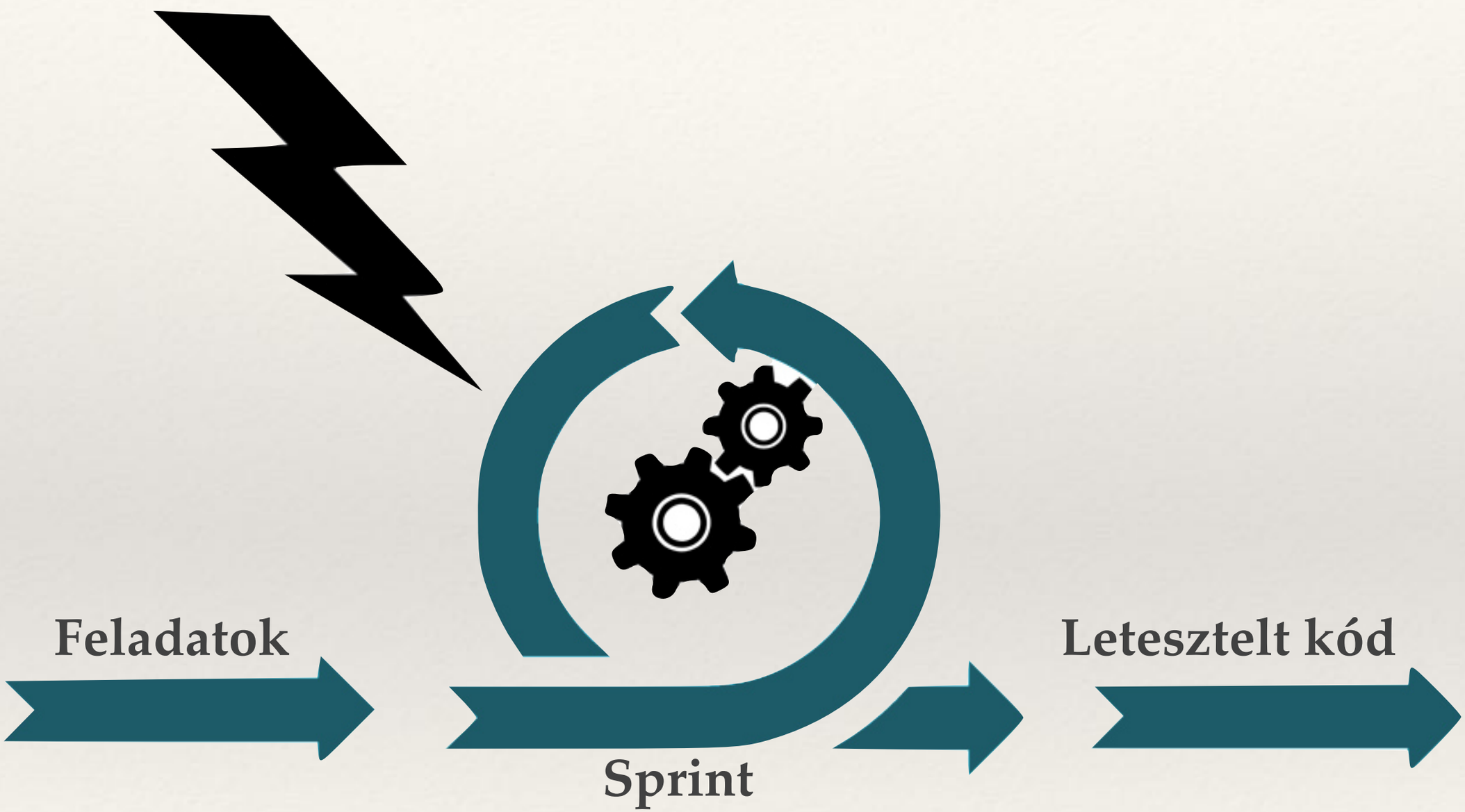
SCRUM alapok

Mit ígér a SCRUM?

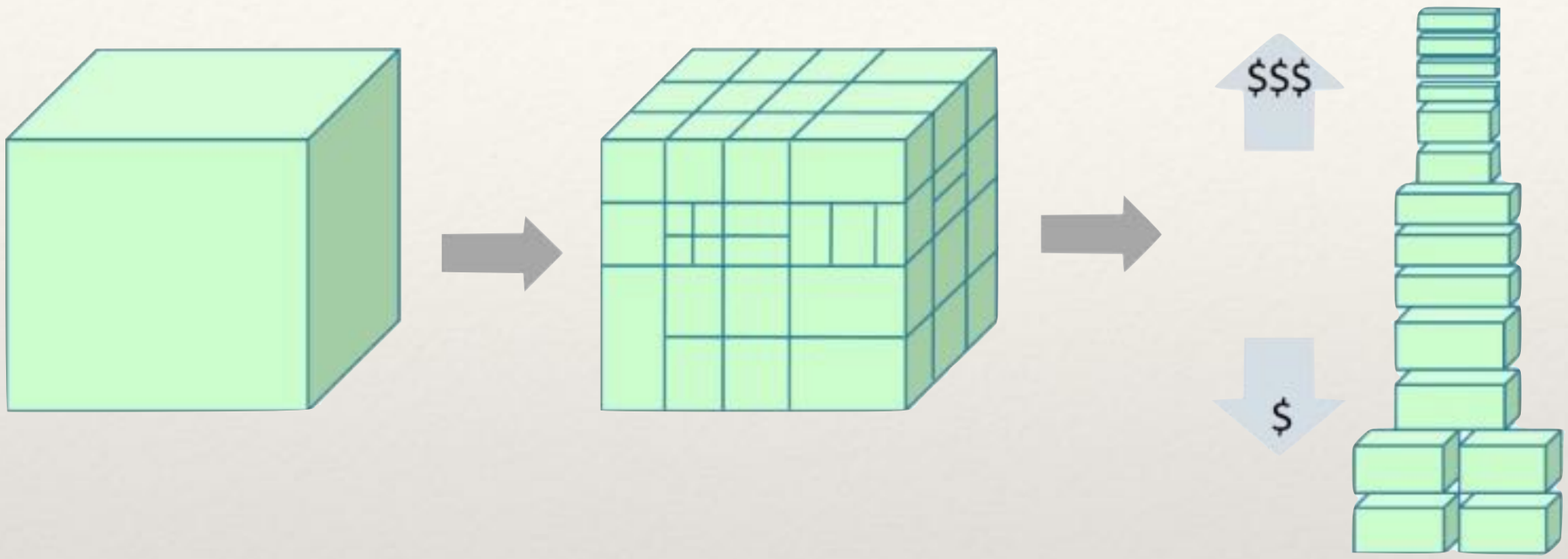


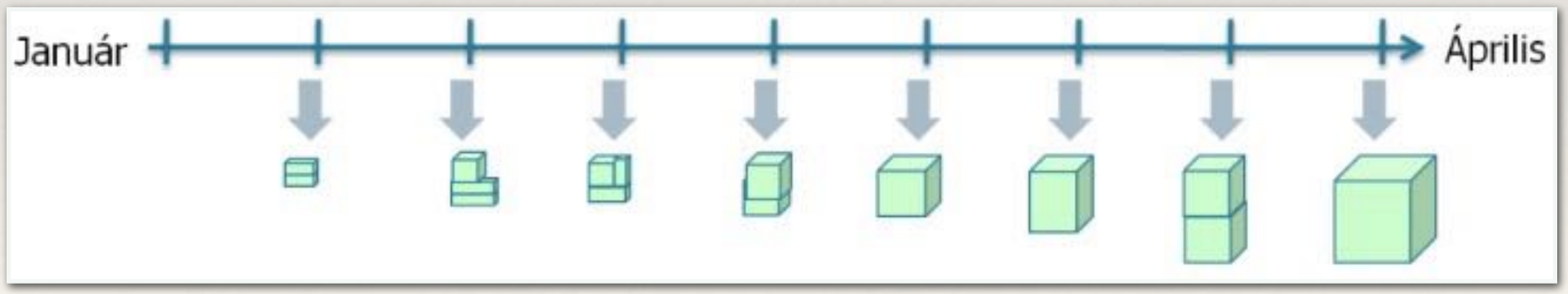
Emich Szabolcs - sz@bolcs.hu

Sprint közben lehetőleg nincs változás



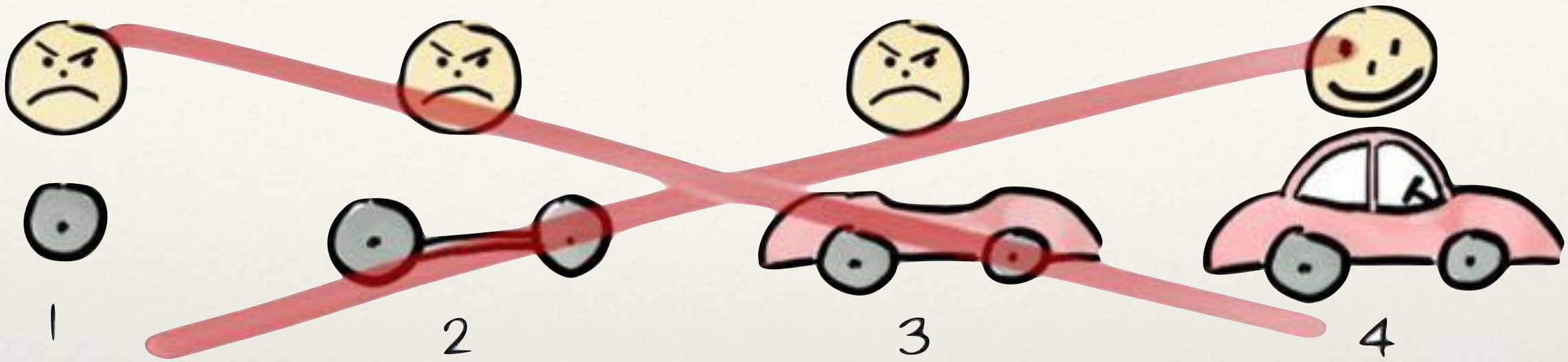
Inkrementális, iteratív



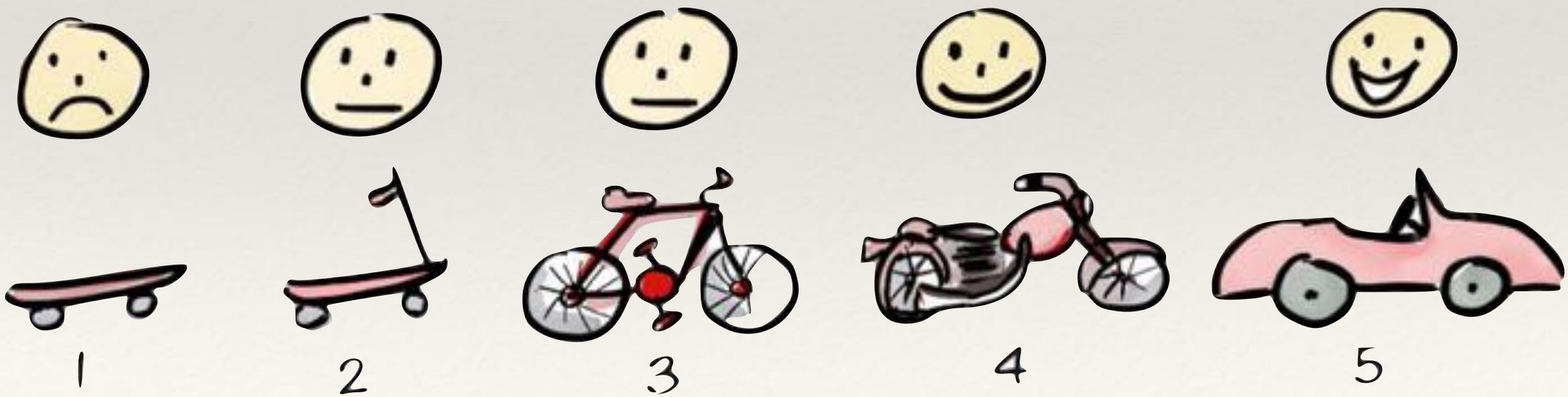


Inkrementálisan

Nem így…



Így!



SCRUM szerepek

Ő képviseli az ügyfél igényeit. Ő felel az eredményekért.

Önirányító módon működik és felel a termék inkrementumért

Csapat

Emlékeztet a szabályokra és segít elhárítani az akadályokat

- Termékfunkciókat definiálja, megválaszolja a termékkel kapcsolatos kérdéseket
- A backlogot priorizálja az üzleti érték (Business Value) alapján és egyezteti a különböző érdekeket.
- Eldönti, hogy mikor mehet ki a termék élesbe
- Minden sprint elején újragondolja a feladatokat és prioritásokat

- Elfogad vagy visszautasít új funkciókat
- Felel az üzleti sikerért

A PO dönt arról, hogy MI lesz leszállítva

- Elvállalja, hogy a PO által priorizált backlogból a legmagasabb értékű funkciókat megvalósítja
- Becslést tesz arra, hogy mennyit fog tudni leszállítani
- Megszervezi a saját munkáját
- Minden szükséges funkció a csapaton belül van
- A csapattagok ideális esetben állandó tagok, a csapatok összetétele a Sprintek közötti időszakban változhat

A csapat dönt arról, hogy mennyit tud leszállítani

- Biztosítja azt, hogy a csapattagok emlékezzenek a játékszabályokra, vállalásokra, ígéretekre.
- Elhárítja az akadályokat
- Elősegíti azt, hogy a csapat hatékonyan dolgozzon
- Erősíti a project szervezet tagjai közti kapcsolatot, kommunikációt

A SM segít betartani a szabályokat, elhárítani az akadályokat

Klasszikus PM szerepek

- Mi lesz az alábbi kockázatokkal, ki felel értük?
  - Scope
  - Idő
  - Költség
  - Kommunikáció
  - Kockázat
  - Minőség
- Mikor kell PM a három SCRUM szerepen kívül?
- 5 perc megbeszélni

Klasszikus PM szerepek

Mi lesz velük a SCRUM-ban?

Felelősség	Product Owner	Csapat	Scrum Master
Hatáskör	✓ (release)	✓(sprint)	x
Idő	✓ (release)	✓(sprint)	x
Költség	✓	x	x
Kommunikáció	✓	✓ (sprint report)	x
Rizikó	✓	✓	✓
Minőség	✓ (scope)	✓ (tesztelés)	✓ (folyamat)

SCRUM dokumentumok

Product backlog

- Egy priorizált követelmény lista
  - a PO kezeli, de mások is hozzájárulhatnak ötletekkel
- Tartalmazza az igényeket a termékkel kapcsolatban
- A backlog elemek leggyakrabban User Story formátumban készülnek el
- a fejlesztő csapat relatív becslést ad a különböző User Storik komplexitásáról

Mi nem a Product backlog?

- Nem egy teljes lista arról, ami szükséges lehet
- Nem megoldások listája, hanem követelményeké
- Nem egy szerződés ügyfél és a PO vagy a csapat között
- Nem egy tevékenység (TODO) lista

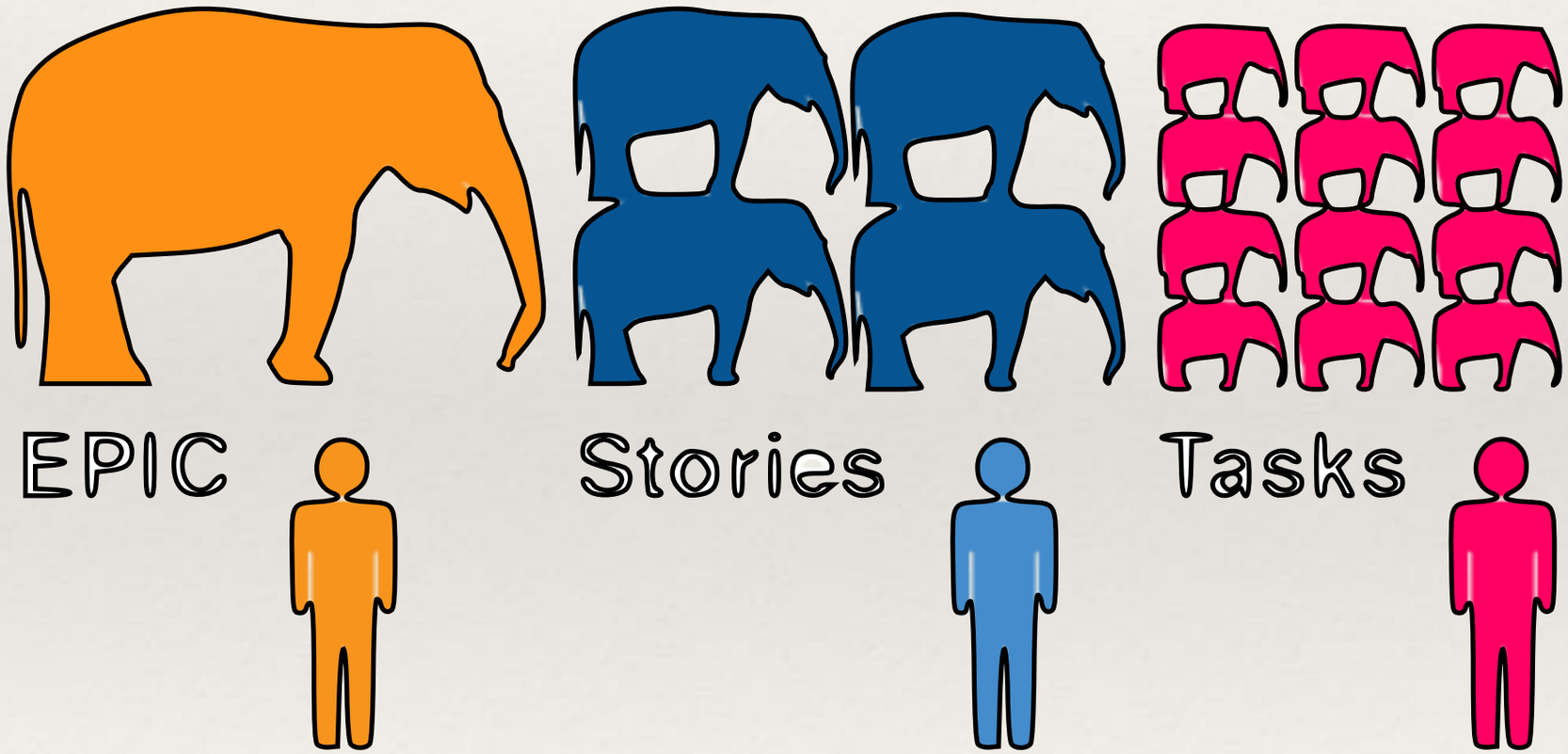
Product backlog és a részletek

- A backlog elején a részletesen kidolgozott, magas prioritású storyk vannak. Készek arra, hogy a következő sprintbe kerüljenek.
- Ezek alatt vannak közepesen kidolgozott storik. Előesztimáláson már átestek, de még nincsenek részletesen kidolgozva.
- Lejjebb haladva elnagyolt, ötletszerű, Epic szintű storykat találunk

Sprint backlog

- a fejlesztő csapat birtokolja
- a karbantartását is ő végzi, de bevonhatja a ScrumMastert is
- tartalmazza a taszkokat, az eredeti becslést és a fennmaradó ráfordítást is
- a csapat frissíti naponta, azért hogy mindig az aktuális, még hátramaradt feladatokat mutassa

Storyk mérete



Több hét v. hónap | 1-5 nap | 1-4 óra

User Story

When ____

I want to ____

So I can ____

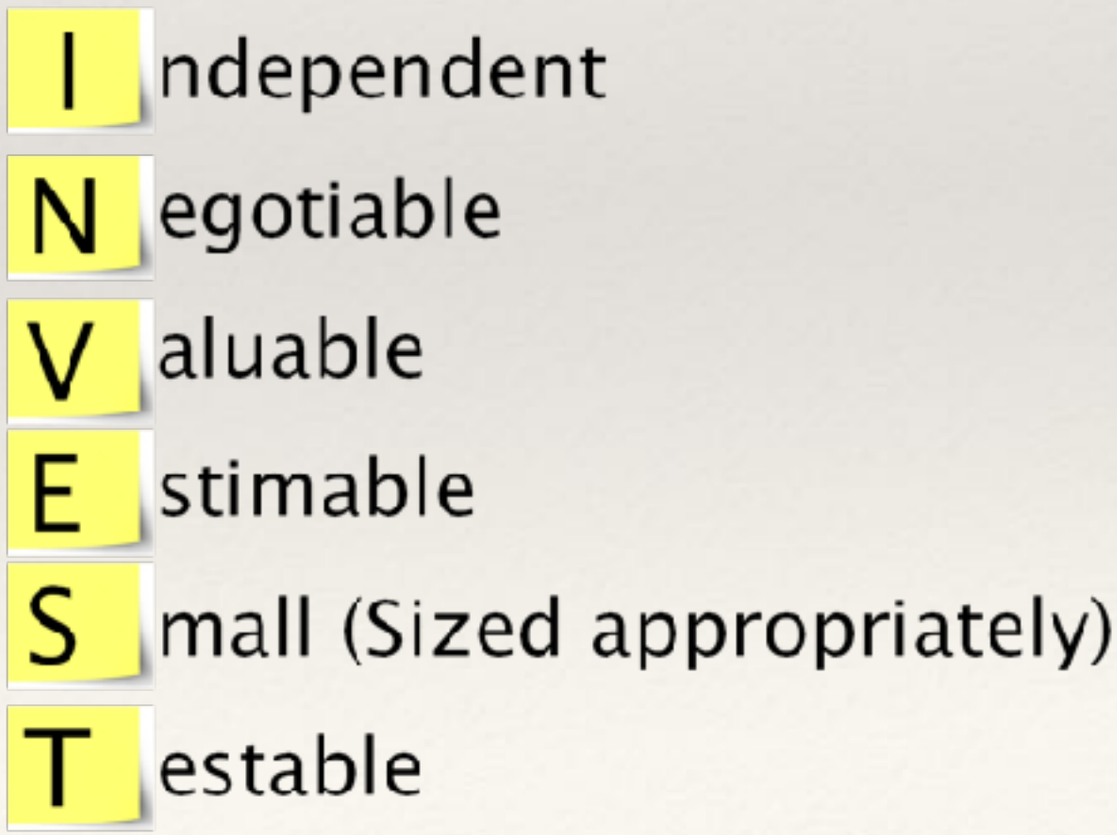
Situation

Motivation

Expected Outcome

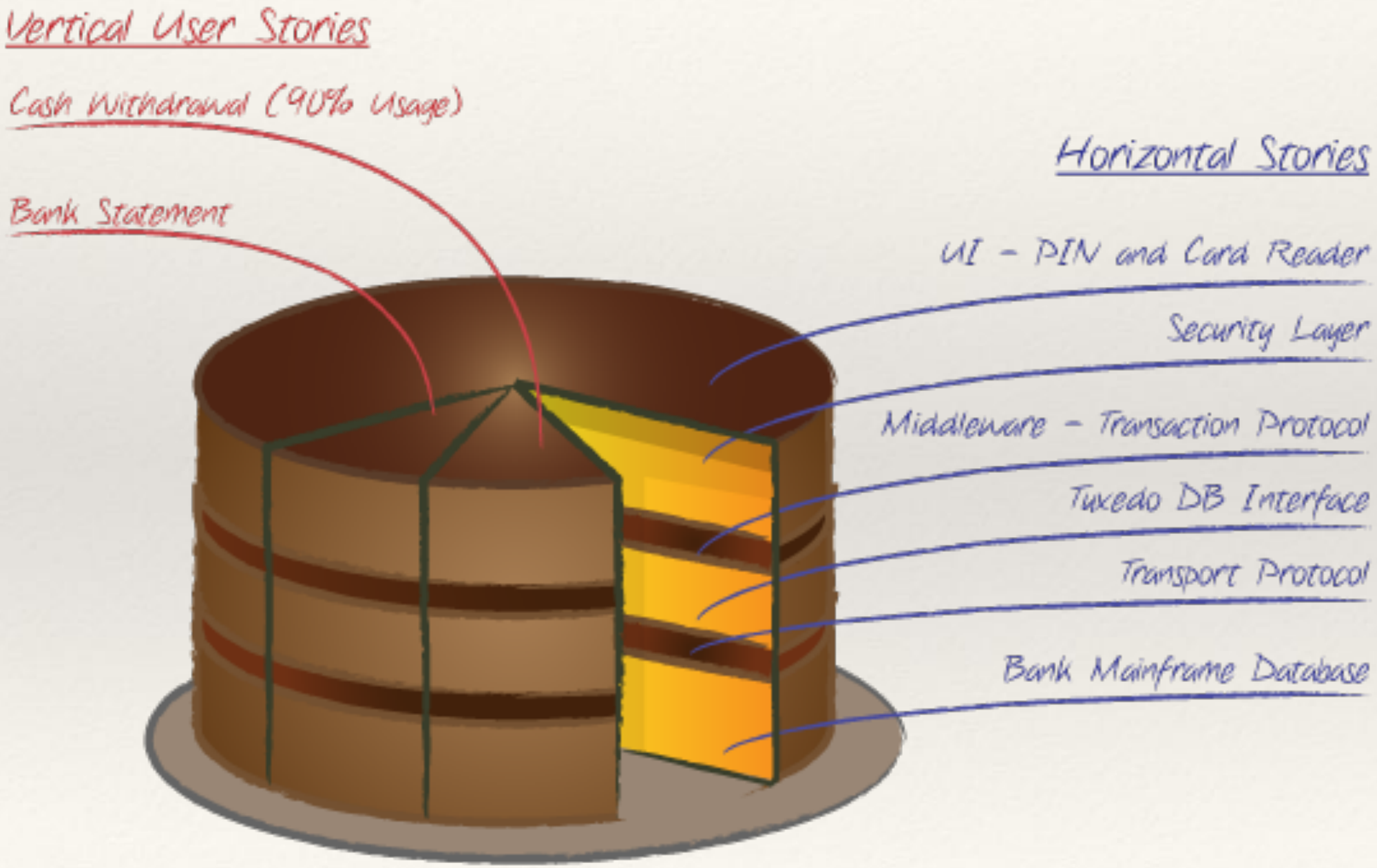
User Story

- természetes nyelvezettel leírja, hogy ki, mit, miért szeretne
- rövid, nem több 2 mondatnál
- nem tartalmazza az összes részletet
- lehet nagyon általános, de nagyon specifikus is
- tartalmaz elfogadási kritériumokat



A User Storyk vertikálisak

Automated Teller Machine (ATM)
Horizontal and Vertical User Stories - Slicing the Cake



Jó Story vagy sem?

- Mint ügyfél szeretném az adataimat rugalmas módon tárolni egy adatbázisban

I ndependent

N egotiable

V aluable

E stimable

S mall (Sized appropriately)

T estable

Jó Story vagy sem?

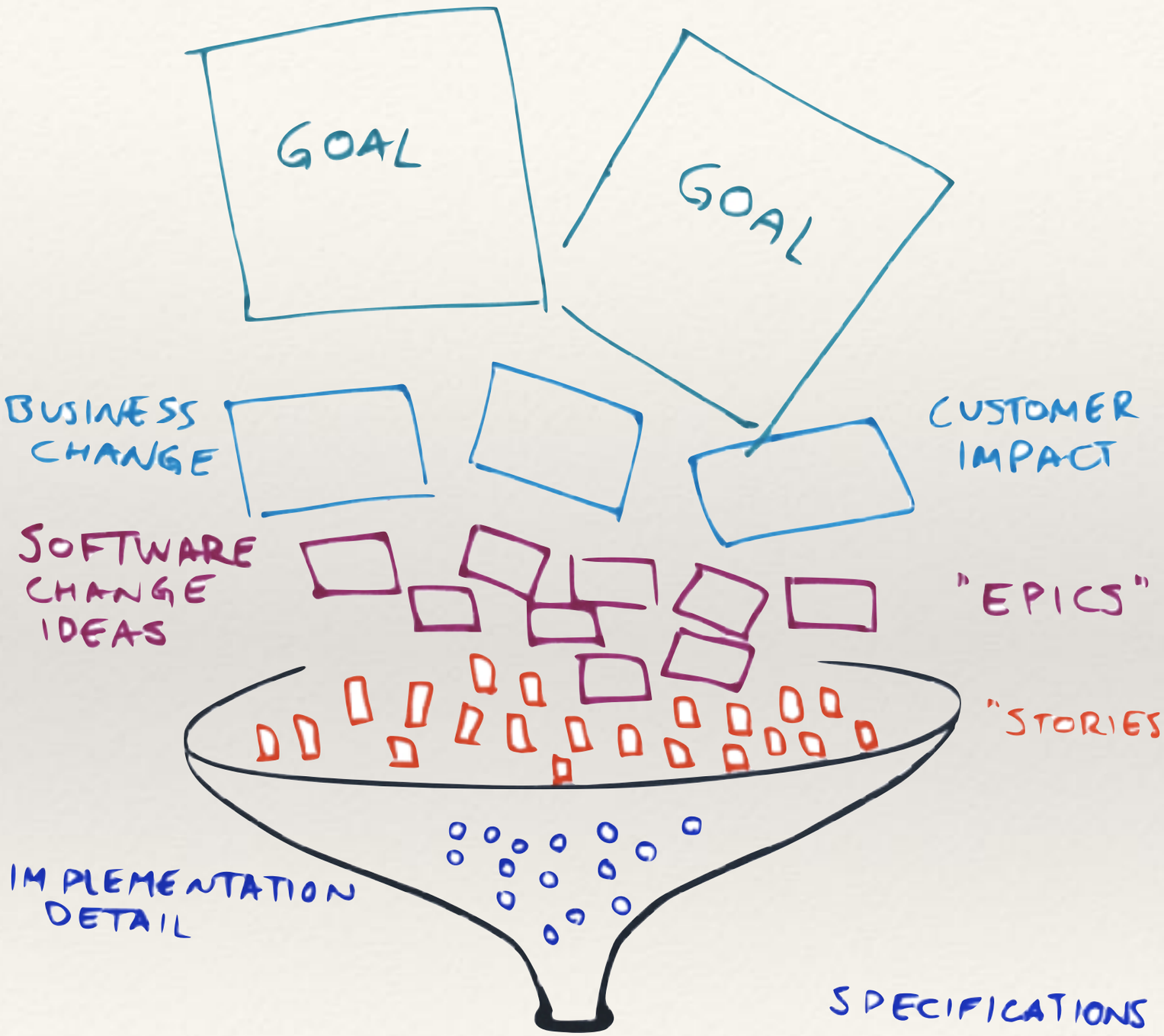
- Üzletkötőként szeretném tudni azt, hogy egy ügyfélnek milyen kedvezmény jár, azért, hogy személyre szabott ajánlatot adhassak
  - elfogadási kritérium:
    - mind ügyfélnél látom, hogy a meglévő kedvezmény szabályaink alapján milyen kedvezmény jár neki
    - új ügyfélnél a standard kedvezmények legyenek
    - késve fizető ügyfélnél jól láthatóan figyelmeztessen a rendszer

Túl nagy storyk szeletelése

1. A workflow mentén - a nagy US valószínüleg több workflowt tartalmaz, ezeket gyakran szét lehet bontani. Minden új US a workflow egy-egy eleme
2. A forgatókönyv szerint. Egy US a fő forgatókönyvnek, és a mellék vagy hiba szálaknak is külön
3. Műveletek szerint - a CRUD műveletek mentén könnyen bonthatóak a storyk
4. Adat típusok, tartalom szerint - a különböző nyelveknek külön storyk, vagy különböző azonosító típusoknak külön user story
5. Az input v. output vagy konfiguráció eltérései szerint
6. Szerep szerinti bontás - ezt támogatja a szokásos story séma is
7. Ismeretek szerint - a story jól feltárt és megtervezett része marad egy storyban, az ismeretlen részek meg mennek egy spikeba pl.
8. Komplexitás szerint - kis komplexitású rész külön, majd az egyre nagyobbak...

Tippek és trükkök

1. Használj hierarchikus storykat



Astronauta plans

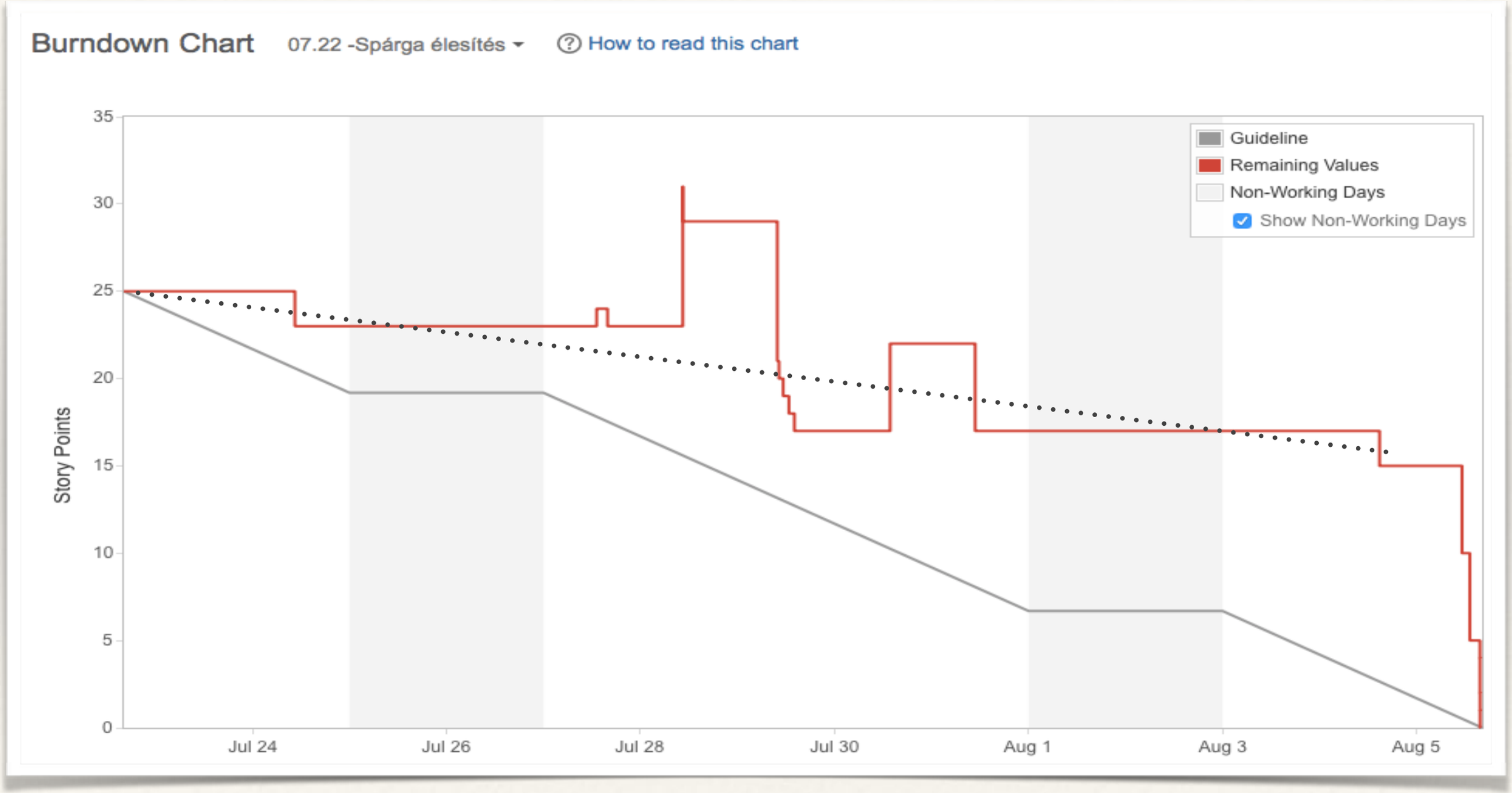
Tippek és trükkök

2. A viselkedésbeli változásról szóljon a story
   - próbáld számszerűsíteni a várt változást
   - az 5% -al gyorsabb is érték (storyk bontása)

Szeretném a nagy fileokat is gyorsan feltölteni (20%-al gyorsabban)

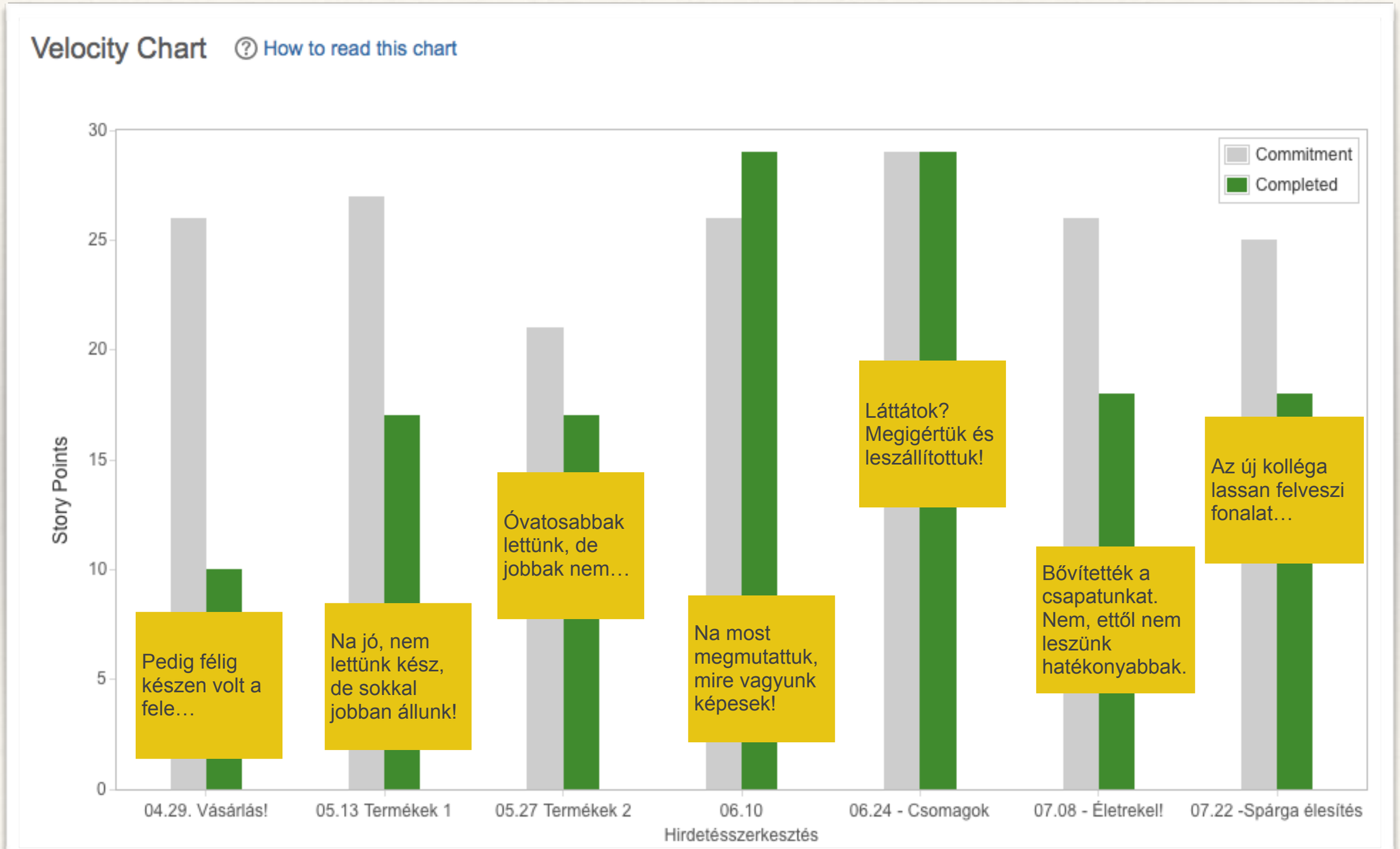
További tippek:
Gojko Adzic: 50 quick ideas to improve your User Stories

Sprint burndown

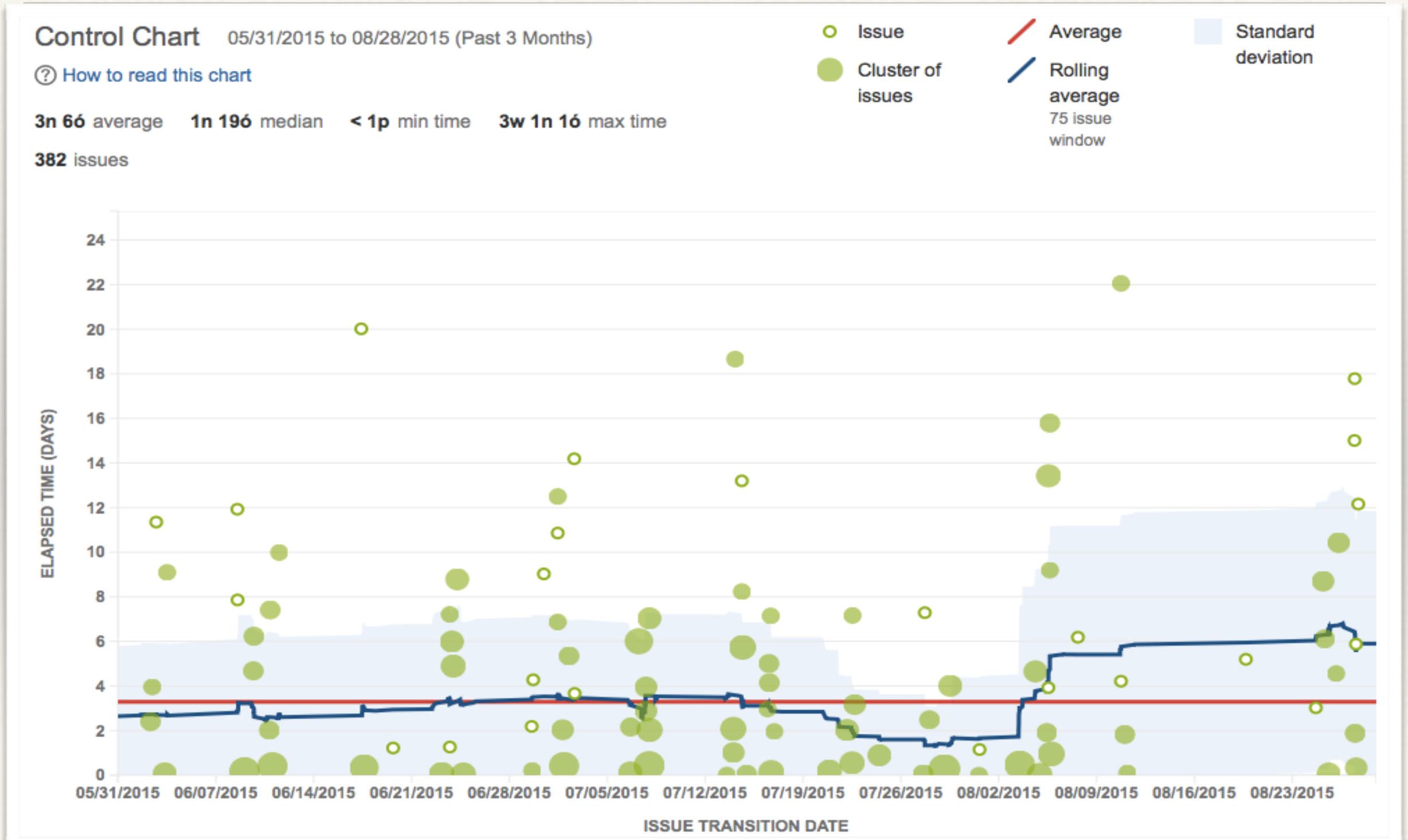


a csapat haladását vizualizálja

Velocity chart



Control chart



Definition of Done (DoD)

- olyan ellenőrzési kritériumok felsorolása, amik minden storyra érvényesek
- a csapat élete során, ahogy éretebbé válik, egyre hoszabb lesz a Dod
- nem csak technikai feltételeket tartalmaz

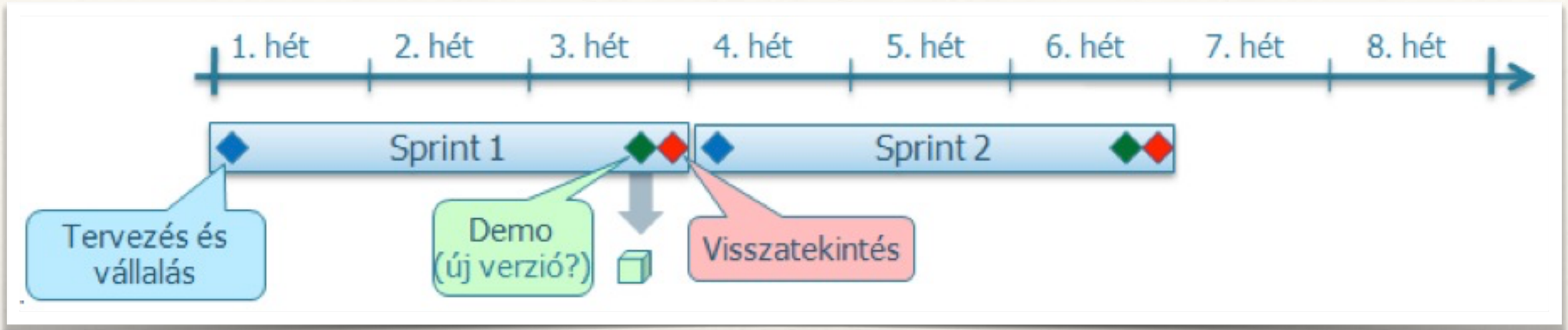
Planning

Stand-Up

Review

Meetingek

Retrospektív


1. hét
2. hét
3. hét
4. hét
5. hét
6. hét
7. hét
8. hét
Sprint 1
Sprint 2
Tervezés és vállalás
Demo (új verzió?)
Visszatekintés


1. hét
2. hét
3. hét
4. hét
5. hét
6. hét
7. hét
8. hét
Visszatekintések(4h)
Tervezés (igény esetén)
Verzió adás (igény esetén)

Sprint planning

az új iteráció hivatalos elindulása

- A meeting előtt
  - a csapat, a PO segítségével esztimálja a feladatok komplexitását
  - a PO priorizálja a feladatokat
  - megtervezni a csapat kapacitást
- A meeting közben
  - a csapat az eddigi teljesítményét vizsgálva eldönti mennyit tud bevállalni
  - a PO és a csapat megbeszéli az esetleges kérdéseket a feladatokkal kapcsolatban
  - a csapat a legfontosabb feladattal kezdve taszkosítja a storykat
- Meeting után
  - a ScrumMaster elindítja a sprint backlogot

Csapat kapacitás

sprint planning előtt

- példa a csapat kapacitás kalkulációra
  - teljes kapacitás: 100%
  - SCRUM overhead (meetingek): 10%
    - előre tervezés, előesztimálások: 10%
    - nagy prioritású support feladatok: 25%
    - a sprint céljainak elérésére marad: 55%
- ezen felül érdemes még figyelembe venni, hogy mennyi lesz a kieső embernapok száma szabadságok, vagy ünnepek miatt

Daily scrum / standup

Csapat tájékoztató és energia

- Célja
  - minden csapattagnak tudnia kell, mit csinálnak a többiek
  - ezt úgy érik el a csapattagok, hogy beszélnek a munkájukról
- Hogyan érhető ez el?
  - 3 kérdésre kell válaszolni:
    - Mit csináltál az előző találkozó óta
    - Mit fogsz ma csinálni?
    - Milyen akadályokat, problémákat látsz?
- A daily scrum után
  - a ScrumMaster az akadályok elhárításán dolgozik

Sprint review/Demo

Az igazság pillanata, visszajelzés

- A demo előtt
  - a ScrumMaster átnézni, hogy mely feladatok lettek teljesen befejezve és ez alapján kiszámítja a Velocity-t
  - a csapat felkészül a demóra, csak az elkészült új funkciókat fogják bemutatni
- A demo során
  - a csapat bemutatja az elkészült és letesztelt új funkciókat
  - a PO elfogadja, vagy elutasítja az eredményt
- A demo után
  - a PO és a ScrumMaster frissíti a Product Backlogot, valamint felkészülnek az új sprintre

Retrospektív

Csapat belső tanulási lehetősége

- Célja
  - A csapat lehetőséget kap hogy átgondolja, miként javíthatná a munkát
- Ki vesz rajta részt?
  - A fejlesztési csapat, a ScrumMaster és a PO
  - Mások csak akkor, ha a csapat kéri
- Mire nem való a retrospektív?
  - teljesítmény értékelésre
  - személyeskedésre, bűnbak keresésre

Esztimálás

Story pont

- relatív érték
- csökkenti a bizonytalanságot
- a feladat komplexitását, a ráfordítás mértékét mutatja
- meghatározása gyors
- nem avul el és nem függ a tapasztalattól
- a csapat intelligenciára, tapasztalatra támaszkodik

Planning poker

- mindenki részt vesz, és hozzáadja a tudását
- Fibonacci skála (1, 2, 3, 5, 8, 13, 20, 40)
- az esztimálás egyidőben történik
- a kilógó értékeket megbeszéljük
- addig folytatjuk, amíg nincs egyetértés
- korábbi tapasztalatok segítik
- sok szempontból meg lehet vizsgálni egy problémát
- gyors eredmény

Story pont

- nincs mértékegysége
- relatív érték a többi becsléshez képest
- a feladat komplexitását, méretét mutatja
- meghatározása gyors
- nem avul el és nem függ a tapasztalattól
- a csapat intelligenciára, tapasztalatra támaszkodik
- Velocity = egy sprintben szállított Story pontok száma (általában az utolsó 3 sprintre átlagolva)

Business value

- egy funkció üzleti értéke az ügyfél szempontjából, a többi backlogban lévő funkció értékéhez képest
- termékfejlesztés esetén segít a prioritás kialakításában

SELECTION CRITERIA ADD CRITERIA		Solr		MapsDB		Licit könnyíté:	
	Weight	Rating	Weighted Score	Rating	Weighted Score	Rating	Weighted Score
Q4 bevétel	5	2	10.00	1	5.00	5	25.00
Ügyfél elégedettség	4	4	16.00	2	8.00	3	12.00
Piac növelés	1	1	1.00	4	4.00	4	4.00
Stabilitás	4	5	20.00	2	8.00	1	4.00
Net Score			47.00		25.00		45.00
Rank			1		3		2

Microsoft, 1978

A csapat

Ideális esetben

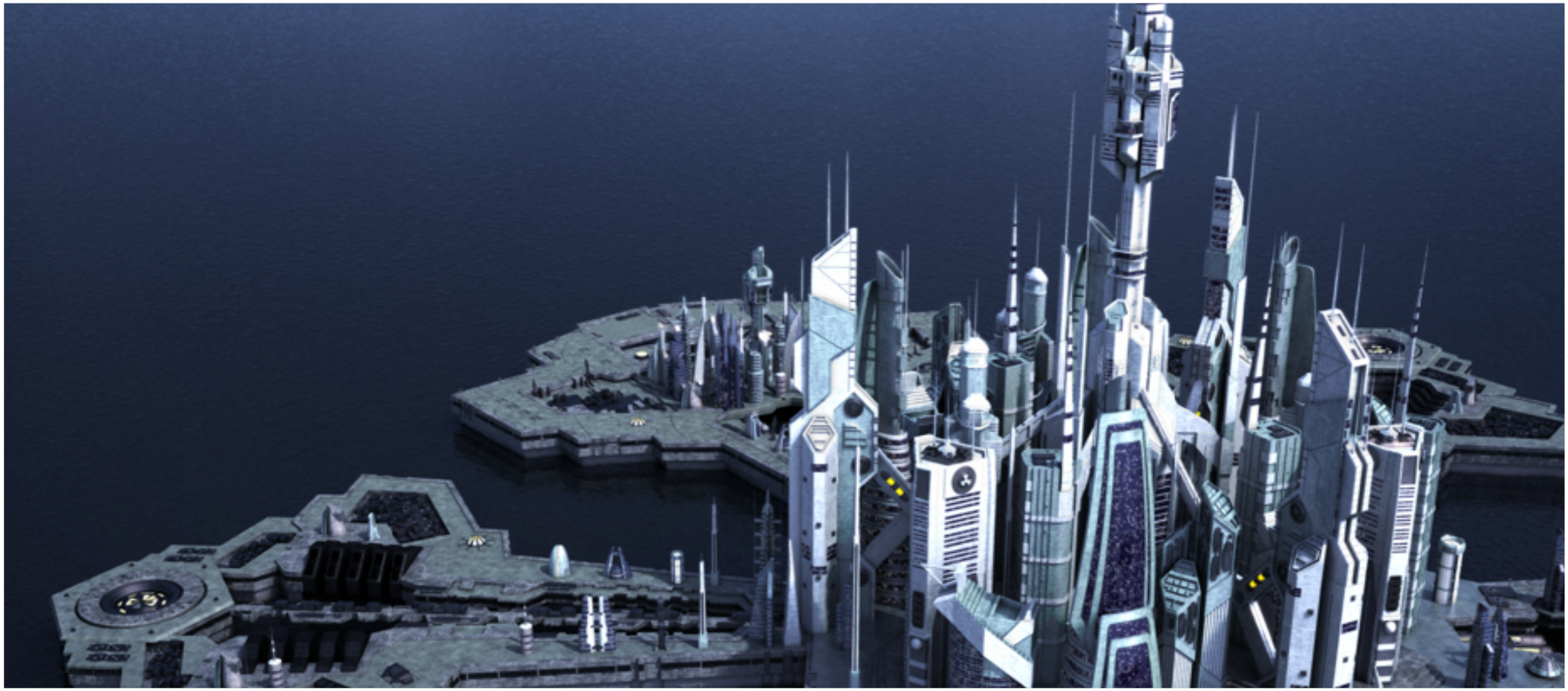
- Felhatalmazott és önálló
- Megszervezi magát és a munkáját
- Mérete 7+/- 2
- Keresztfunkcionális - minden tudás megvan a csapatban ahhoz, hogy kész végeredényt állítson elő
- Teljes munkaidőben együtt dolgoznak
- Elkötelezett a sprint célért
- Közös felelősséget vállal

Multitaszking

- Vonzó lehetőség, hogy több dolgon dolgozzunk egyszerre
- Mégis jobban tesszük ha nem próbálunk multitaszkolni
- Az idő 20% megy el a kontextus váltásra
- Ha a feladatokat lépésenként oldjuk meg, sokkal előbb van visszajelzésünk a már elkészültekről
- Ha minimalizáljuk a folyamatban lévő dolgok számát, maximalizálni tudjuk az áteresztő képességet

Csapat fázisok

- **Forming**
  - sok energia, izgalom
- **Storming**
  - szerepek, képességek összecsapása
  - fogalmak tisztázása, értelmezése
  - klikkek kialakulása (ők és mi)
- **Norming**
  - kiégettség érzés a storming után
  - időnként feszültség a beszélgetésekben
  - felelősségvállalásban óvatosság
  - kevés önállóság
- **Performing**
  - a csapat megismerte erősségeit
  - kialakul a folyamatos tanulás szokása

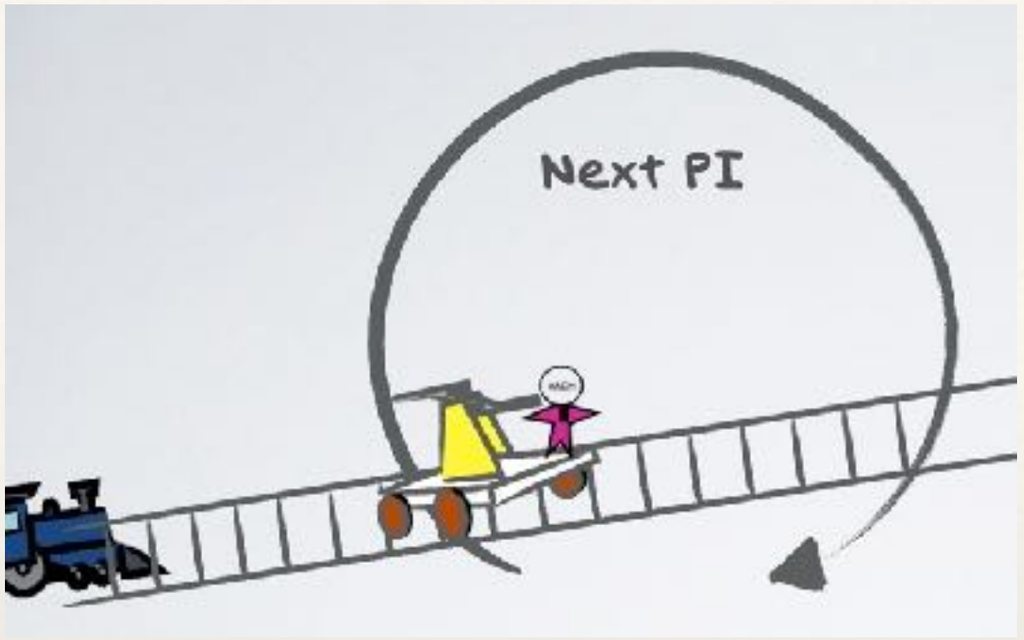


Skálázható agilitás

Keretrendszerek

- **SoS (SCRUM of SCRUMs):** néhány SCRUM csapat esetén, könnyen bevezethető
- **LeSS**: SCRUM továbbfejlesztve (8 csapatig)
- **LeSS Huge**: SCRUM sok ezer emberrel
- **SAFe**: Agilitás skálázva, részletesen kidolgozott
- **Nexus:** a scrum.org keretrendszere

SAFe



- **Új szintek**: csapat, program, value stream, portfolio
- **Új szerepek:** solution manager, value stream engineer, solution architect, enterprise architect, release train engineer (SM)
- **Új meetingek**: scrum of scrums, integral system demo, program board
- **Új folyamatok**: ART - agile release train, 4 program increment utan 5. innovation planning

Scaled Agile Framework®

SAFe®

PORTFOLIO VISION

Program Portfolio Management

Portfolio Metrics

Strategic Themes

Epic Owners

Enterprise Architect

Kanban

Portfolio Backlog

NFRs

Business Epic

Arch. Epic

Business Epic

AGILE RELEASE TRAIN

AGILE RELEASE TRAIN

AGILE RELEASE TRAIN

Budgets

Lean

Lean-Agile Leaders

Epics span releases

Architecture evolves continuously

Coordination

- Content
- Integration
- Releasing

Value Streams deliver solutions

PORTFOLIO

ART Metrics

Release Management

AGILE RELEASE TRAIN

Shared

DevOps

UX

System Architect

RTE

Release on Demand

Program Epics

Business Owners

Program PI Objectives

Product Management

Vision

Roadmap

Program Backlog

NFRs

WSJF

Release Planning

Feature

System Demo

Feature

Arch

I&A

WSJF

Release Planning

Program Increment

Feature

Feature

I&A

WSJF

Release Planning

Program Increment

Features fit in releases

Architectural Runway

System Team

Develop on Cadence

Team Backlog

NFRs

Team PI Objectives

Plan

EXE

Demo & Retro

I P

Sprint Goals

Stories fit in iterations

Team Backlog

NFRs

Team PI Objectives

Spikes, Refactors, Other

Product Owner

Scrum Master

AGILE TEAMS

Developers & Testers

Code Quality

- Agile Architecture
- Continuous Integration
- Test-First

Iterations

Iterations

PROGRAM

TEAM

v 3.0

LeSS

QUEUEING THEORY

LARGE-SCALE
SCRUM IS SCRUM

TRANSPARENCY

MORE
WITH LESS

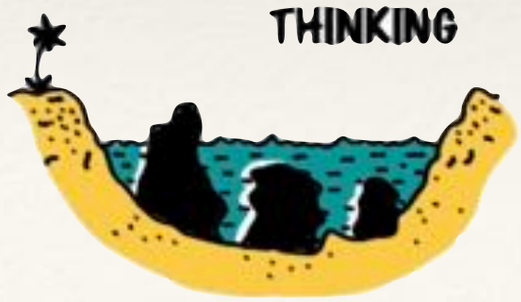
EMPIRICAL
PROCESS CONTROL

SYSTEMS
THINKING

WHOLE
PRODUCT
FOCUS

CONTINUOUS IMPROVEMENT
TOWARDS PERFECTION

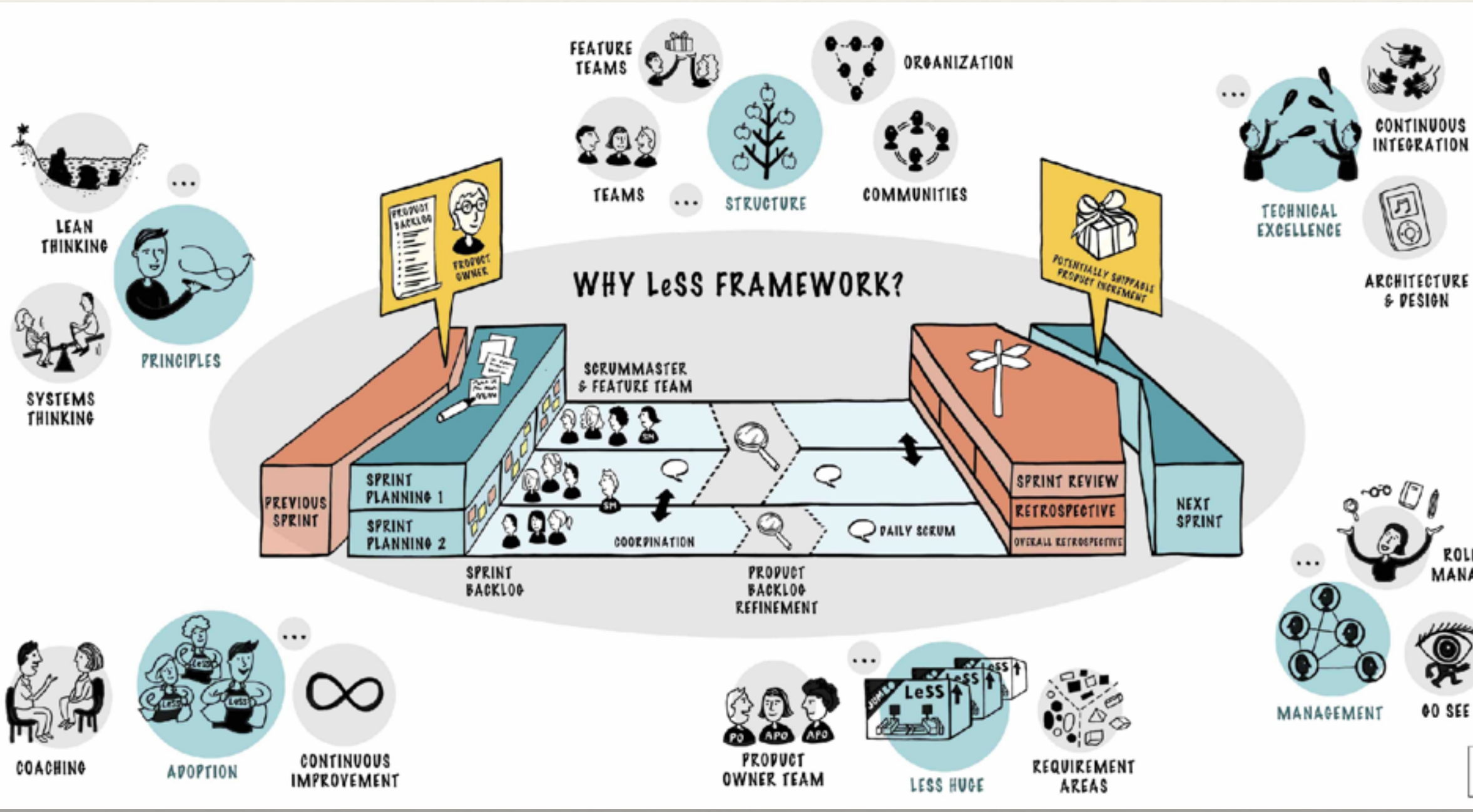
LEAN
THINKING



CUSTOMER
CENTRIC

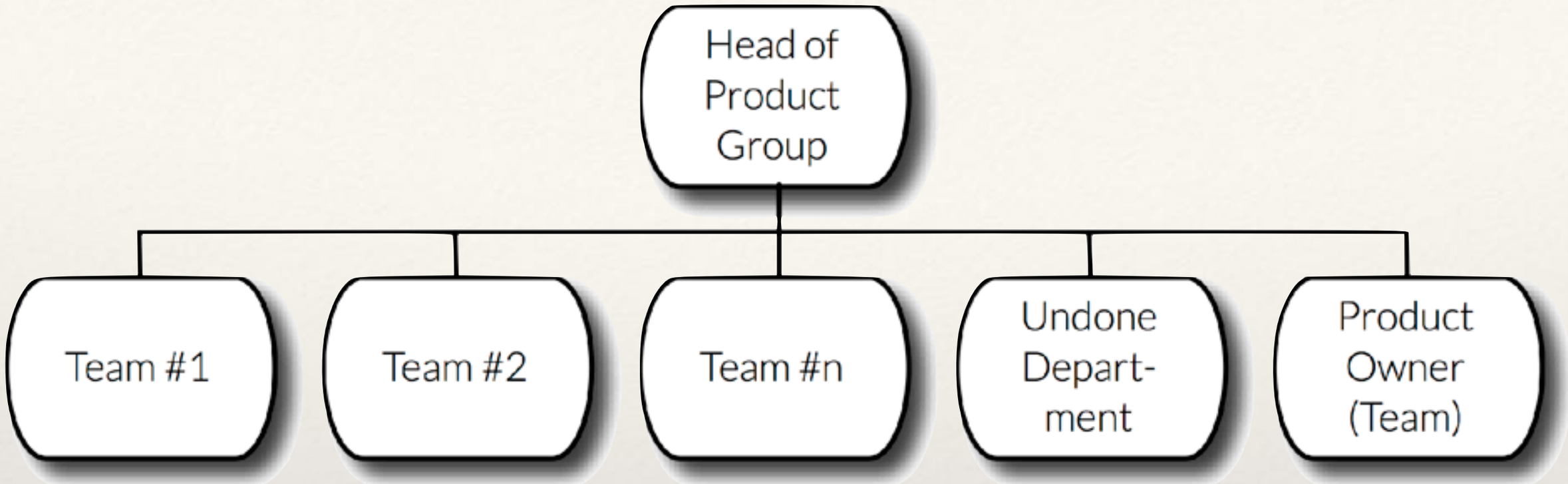
Emich Szabolcs - sz@bolcs.hu

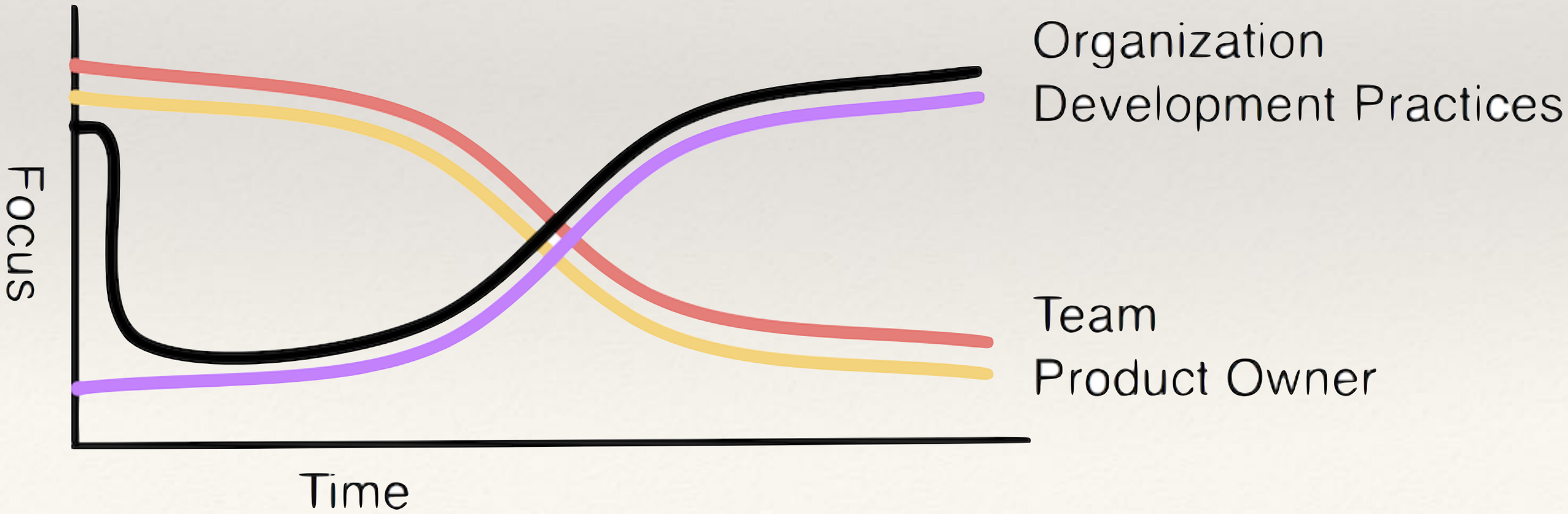
LeSS



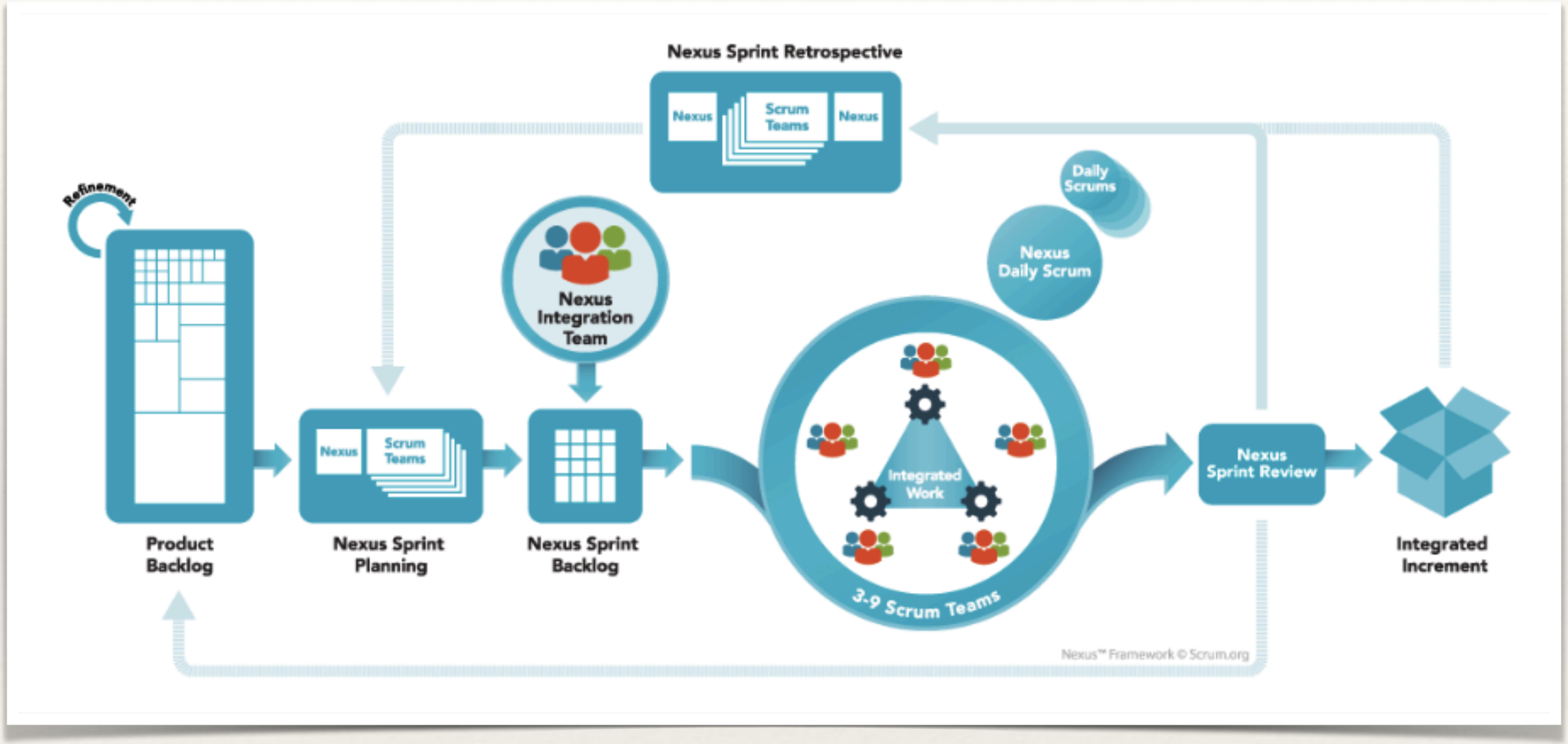
Emich Szabolcs - sz@bolcs.hu

LeSS

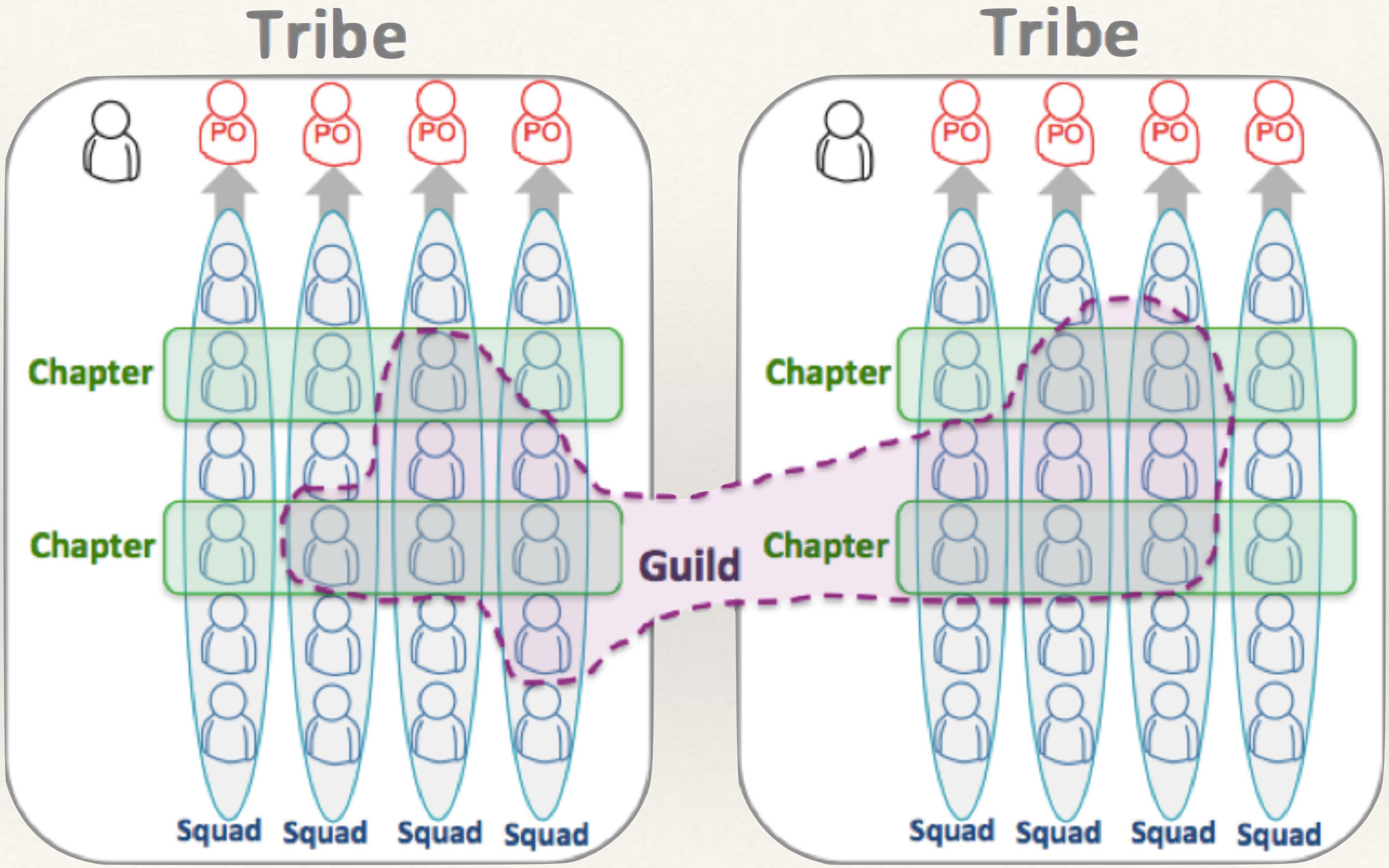

Head of Product Group
Team #1
Team #2
Team #n
Undone Depart-ment
Product Owner (Team)


Organization Development Practices
Team Product Owner
Focus
Time

Nexus


Nexus Sprint Retrospective
Nexus
Scrum Teams
Nexus
Refinement
Daily Scrums
Nexus Daily Scrum
Nexus Integration Team
Nexus
Scrum Teams
Integrated Work
3-9 Scrum Teams
Nexus Sprint Review
Product Backlog
Nexus Sprint Planning
Nexus Sprint Backlog
Integrated Increment
Nexus™ Framework © Scrum.org

Spotify Agile



Kanban

Kanban szabályok