

# Szoftvertchnológia

9-10 előadás: Támogató folyamatok

BSc kurzus

Dr. Balla Katalin




# Tartalom

- Konfigurációmenedzsment, verziókezelés, változáskezelés
- Kockázatmenedzsment
- Minőségmenedzsment
- Mérés és elemzés



# Tartalom

- Konfigurációmenedzsment, verziókezelés, változáskezelés 
- Kockázatmenedzsment
- Minőségmenedzsment
- Mérés és elemzés



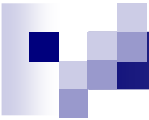
# Konfigurációmenedzsment

- Célja a keletkező munkatermékek integritásának biztosítása, felhasználva az alábbiakat:
  - ☐ Konfiguráció azonosítása
  - ☐ Konfiguráció ellenőrzése
  - ☐ Konfiguráció állapotának követése
  - ☐ Konfigurációs auditok



# CM tevékenységek

- A kiválasztott munkatermékek konfigurációs elemeinek azonosítása
- A konfigurációs elemek változásának követése
- Az alapverziók (baselines) integritásának biztosítása
- Pontos képet adni a konfiguráció aktuális állapotáról a fejlesztők, végfelhasználók és vevők számára
  - A konfigurációmenedzsment alá volt termékek tartalmazzák a felhasználónak átadott elemeket, a kiválasztott belső munkatermékeket, a megvásárolt termékeket, eszközöket és egyéb munkatermékeket, amelyeket az előbbiek létrehozására és leírására használnak.

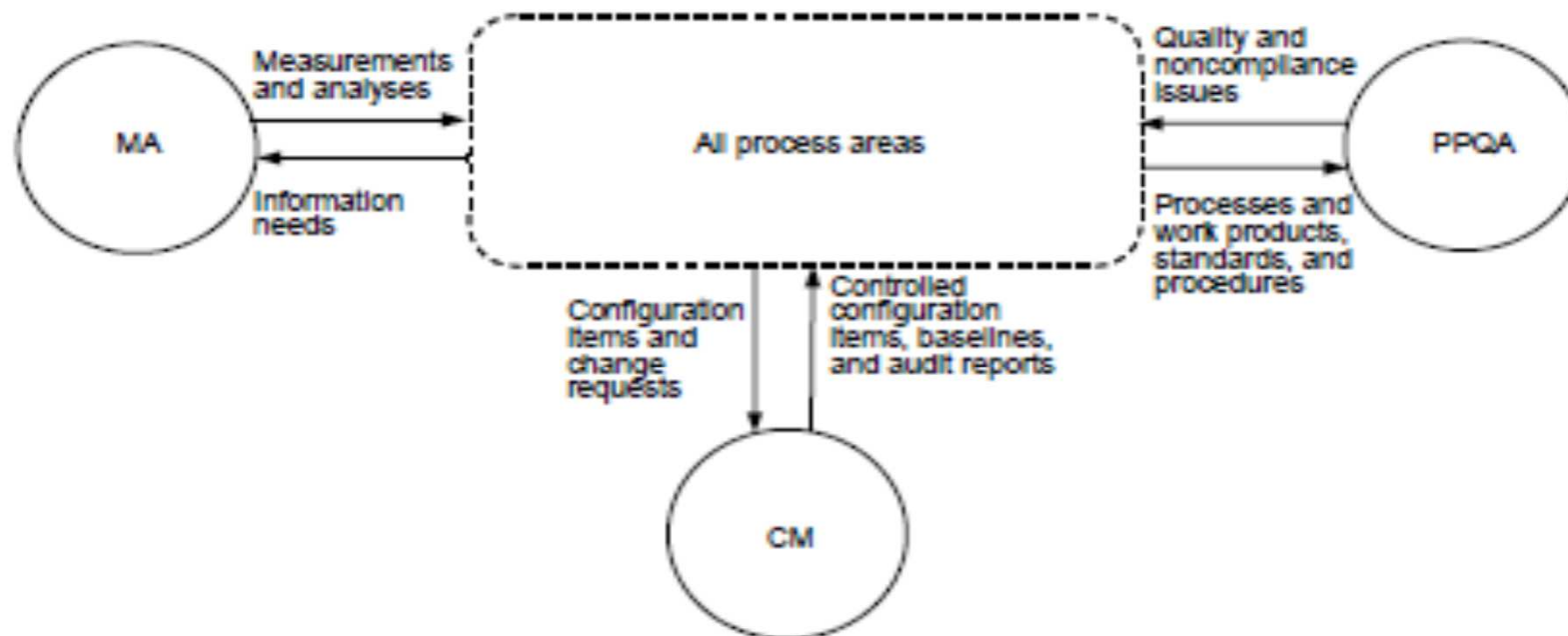


# Mit kell konfigurációkezelés alá vonni?

- Mindent, ami fontos !
- Példák CM alá vonható elemekre:
  - Hardware és eszközök
  - Termék specifikációk, rajzok
  - Szoftvereszközök
  - Kód, könyvtárak
  - Fordítók
  - Tesztelési eszközök, tesztszkriptek
  - Installációs naplók
  - Tervek
  - User story-k
  - Folyamatleírások
  - Követelmények
  - Architektúra tervek, design elemek
  - ...

# CM a CMMI-ben

- Az összes többi folyamatot támogatja!





# CM a CMMI-ben

- SG 1 Alapkonfigurációk létrehozása
  - SP 1.1 Konfigurációs elemek azonosítása
  - SP 1.2 Konfigurációmenedzsment rendszer létrehozása
  - SP 1.3 Alapkonfigurációk létrehozása vagy kibocsátása
- SG 2 Változáskövetés- és ellenőrzés
  - SP 2.1 Változáskérések követése
  - SP 2.2 Konfigurációs elemek felügyelete
- SG 3 Integritás biztosítása
  - SP 3.1 Konfigurációmenedzsment - feljegyzések készítése
  - SP 3.2 Konfigurációs auditok végrehajtása





# CM: Alapverziók / Baselines

- Az alapverziók biztosítják, a konfigurációs elemek stabil alapokkal rendelkezzenek a továbbfejlesztéshez
  - Példa alapverzióra: a termék egy jóváhagyott leírása, amely tartalmazza az egymás között konzisztens követelményleírásokat, követelménykövetési mátrixokat, design-t , egyéb, környezetfüggő elemeket és a végfelhasználói dokumentációt is.
- Az alapverziókat – amint létrehoztuk őket – hozzáadjuk a konfigurációmenedzsment rendszerhez.



# CM: CCB

- „Configuration control”:
  - A CM eleme, amely a konfigurációs elemek formális azonosítása után bekövetkező változásokat elemzi, koordinálja, jóváhagyja vagy elutasítja, és a végrehajtott változások implementálást követi.
- Configuration control board
  - A konfigurációs változások kezelésével megbízott csoport.



# Verziókezelés

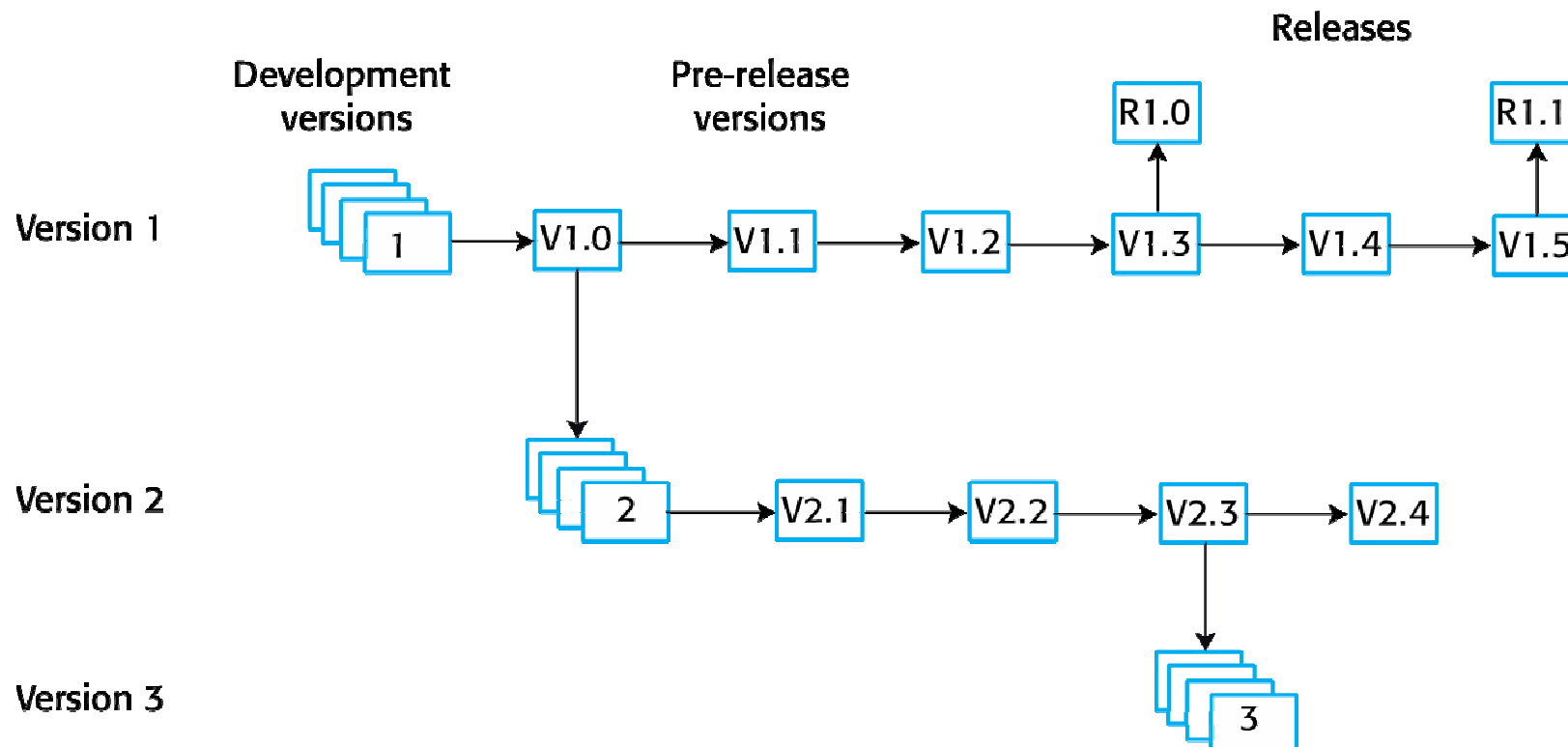
- A CM része
- A verziókezelő rendszerek (Version control (VC)) a komponensek különböző tárolt verzióihoz való hozzáférést szabályozzák.
  - A verziókat általában számokkal és betűkkel azonosítjuk.  
Pl. v1.2
  - Minden verzióhoz időbélyeg tartozik, és nyilvántartják a változást végrehajtó személyt is.
  - A változásokat össze lehet hasonlítani, régebbi verziókat vissza lehet állítani és különböző file-okat össze is lehet merge-ölni.
- Lehetőleg automatizáljuk!

# Verziókezelés

- Többverziós rendszerfejlesztés
- (Multi-version system development )

(Sommerville: Software Engineering, Tenth Edition, ch. 25,

<https://www.slideshare.net/software-engineering-book/> )

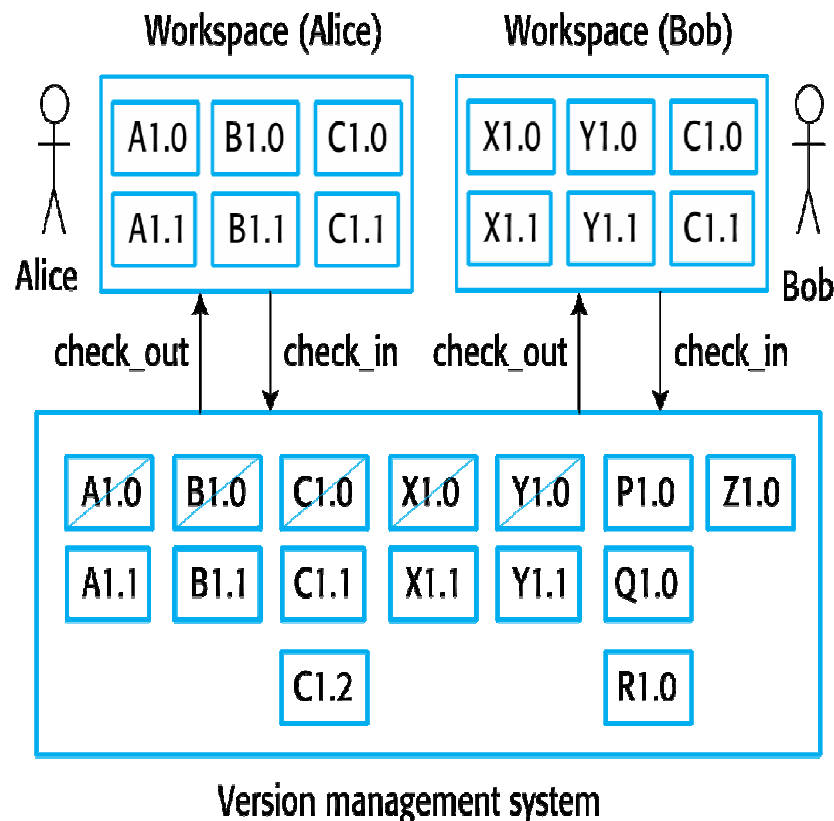




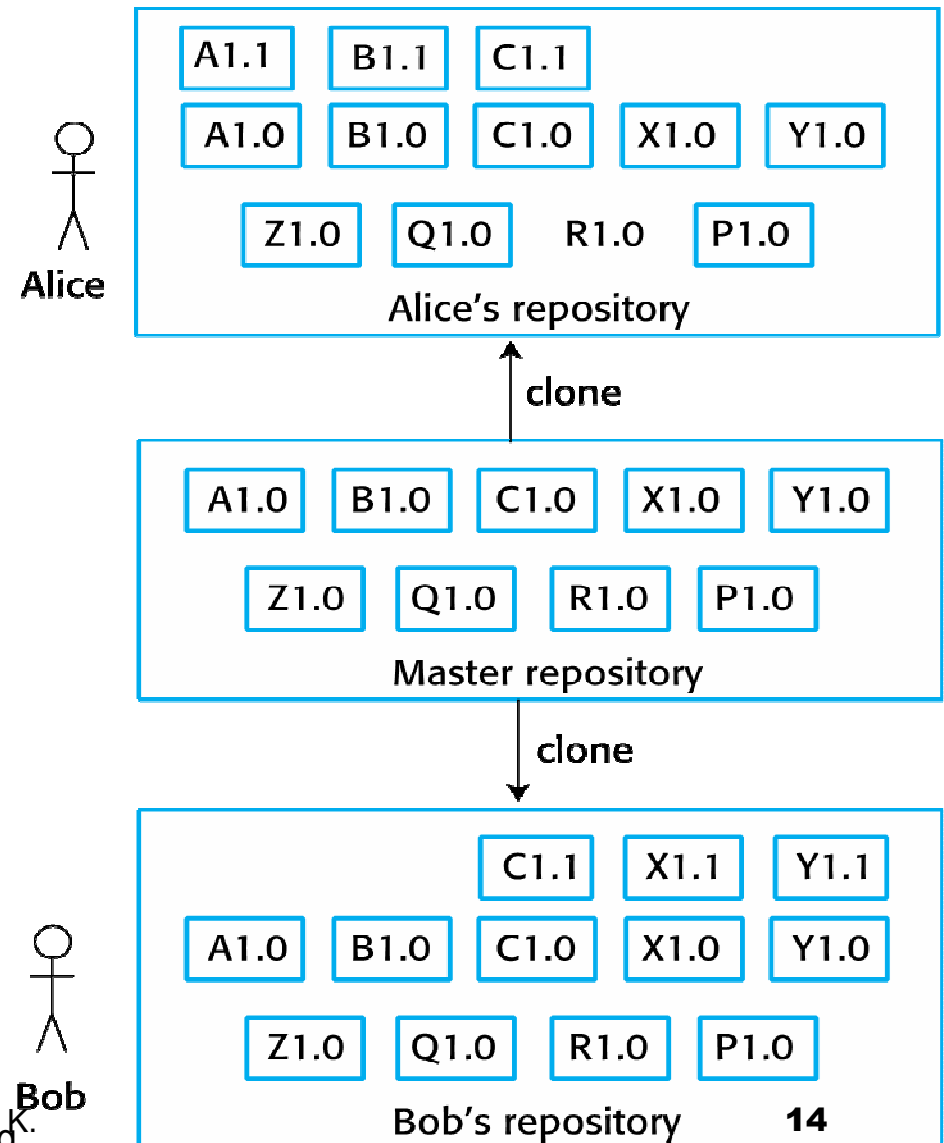
# Verziókezelés

- Kétféle modern verziókezelő rendszert tartanak számon
  - **Központosított rendszerek**, ahol egyetlen „master” repository tárolja a készülő szoftver összes verzióját. A **Subversion (SVN)** széles körben elterjedt központosított verziókezelő rendszer.
  - **Elosztott rendszerek**, amelyekben a komponens repository-nak egyszerre több verziója létezik ugyanabban az időpillanatban. A **Git** széles körben elterjedt elosztott verziókezelő rendszer.
    - Sommerville, Software Engineering, Tenth Edition, ch. 25, <https://www.slideshare.net/software-engineering-book/>

# Verziókezelés



Distributed system.  
Repository cloning





# CM agilis környezetben

- Nagyon fontos, mert a , (CM) is important because of the need to support **gyakori változásokat, gyakori (napi) build-eket** , többféle alapverziót és többféle munkakörnyezetet (pl. az egyénekét, a csapatokét és pl. a páros programozás környezetét is) támogatni kell.
- Hasznos:
  - 1) **a CM-et automatizálni** (pl. hozzunk létre szkripteket a build-ekhez, végezzünk automatikus integritás-ellenőrzést stb.) és
  - 2) a CM-et **egységes, szabványos szolgáltatásként építsük fel.**
- A projekt kezdetén az agilis csapat jelöljön ki egy személyt, aki a CM megfelelő végrehajtásáért felelős. Minden iteráció elején a CM támogatást újra meg kell erősíteni. Az agilis csapatban a CM integrálódik a csapat mindennapi munkájába, a lehető legkisebb figyelemelterelést okozva.



# CM agilis környezetben


## ■ Kollektív tulajdon(lás)

- A kód kollektív tulajdonlása az az explicit konvenció, miszerint minden csapattag minden kód-file-hoz hozzáférhet és azt , szükség szerint, módosíthatja . Ezt azért teheti, hogy teljesítsen egy fejlesztési feladatot, vagy egy hibát kijavítson, vagy javítsa a teljes kódminőséget.





# Tartalom

- Konfigurációmenedzsment, verziókezelés, változáskezelés
- Kockázatmenedzsment 
- Minőségmenedzsment
- Mérés és elemzés



# Kockázat és kockázatmenedzsment

## ■ Kockázat:

- Annak a valószínűsége, hogy előre nem látható esemény fordul elő
- a bizonyosság hiánya arra nézve, hogy egy tevékenység eredménye azonos lesz a tervezettel
- annak lehetősége, hogy egy tevékenység elvárt és tényleges eredménye között bizonyos különbség merül fel

## ■ A kockázat fajtái

- Műszaki
- Pénzügyi
- Kereskedelmi
- Erőforrás
- Vállalati
- ...





# Kockázatmenedzsment

- Azoknak a tevékenységeknek az összessége, amelyek elősegítik, hogy egy tevékenység tényleges eredménye a lehető legjobban közelítse meg az elvárt eredményt
- A kockázat mérséklésére szolgáló intézkedések



# Kockázatmenedzsment

## ■ Tevékenységek

- ☐ A kockázat azonosítása
- ☐ A kockázat elemzése
- ☐ A kockázati tényezők fontossági sorrendjének megállapítása
- ☐ A kockázat elhárítását célzó intézkedések számbavétele
- ☐ A megfelelő intézkedések kiválasztása, bevezetése
- ☐ Az intézkedések eredményének követése



# Kockázatmenedzsment

## ■ Lehetséges, a kockázat elhárítását célzó intézkedések

### ☐ Elkerülés

- ne indítsuk azt a tevékenységet, amelynek kimenetele kétséges egy nyilvánvalóan létező, ható kockázati elem miatt

### ☐ Csökkentés

- a tevékenység megkezdése előtt tegyünk lépéseket, hogy az azonosított kockázati elem hatása minimálisra (elfogadható nagyságúra) csökkenjen

### ☐ Kompenzálás

- fogadjuk el a kockázati tényező negatív hatását a tevékenységre, de egyéb tényezőkre figyelve igyekezzünk ezt a negatív hatás elfogadható nagyságrendűre csökkenteni

### ☐ Megegyezés

- tételezzük fel, hogy a kockázati tényező kifejti hatását, és készüljünk fel a negatív hatás kezelésére



# Kockázatmenedzsment a CMMI-ben

- A kockázatkezelés célja a potenciális problémák azonosítása, mielőtt azok bekövetkeznének.
  - SG 1 Kockázatkezelés előkészítése
    - SP 1.1 Kockázatforrások és -kategóriák meghatározása
    - SP 1.2 Kockázati paraméterek meghatározása
    - SP 1.3 Kockázatkezelési stratégia létrehozása
  - SG 2 Kockázatok azonosítása és elemzése
    - SP 2.1 Kockázatok azonosítása
    - SP 2.2 Kockázatok kiértékelése, osztályozása és rangsorolása
  - SG 3 Kockázatok csökkentése
    - SP 3.1 Kockázatmérséklési tervek fejlesztése
    - SP 3.2 Kockázatmérséklési tervek kivitelezése

# Kockázatmenedzsment agilis környezetben

- A *kockázat* olyan valami, ami megjelenhet és váratlan vagy előre nem látható kimeneteket okozhat.
- „Keep it simple!”

Probability	5	5	10	15	20	25
	4	4	8	12	16	20
	3	3	6	9	12	15
	2	2	4	6	8	10
	1	1	2	3	4	5
		1	2	3	4	5

Balla K.

Impact

23



# Kockázatmenedzsment agilis környezetben

- A simple risk register:

- Description of risk: A one- or two-line overview of the risk. It should be simple and easy to comprehend.

- Date identified: Date when the risk was identified.

- Likelihood: Estimated probability of occurrence of the risk.

- Severity: The severity of the risk is assessed based on impact of the undesired outcome.

- Priority (optional): This could be either given an independent value or set as a product of likelihood and severity (above).

A high-severity risk with a high likelihood should receive more importance than a high-severity risk with a low likelihood.

- Owner: The person who manages, controls, and takes action in response to the risk.

- Action: The response defined to manage/control the risk.

- Status: Indicates whether the risk is open or closed or being monitored

- <https://www.scrumalliance.org/community/articles/2013/2013-may/risk-management-in-agile>





# Kockázatmenedzsment agilis környezetben

- Példa


Risk	Probability of Risk	Size of Loss (Days)	Risk Exposure (Days)
Backup and restore may require the inclusion of additional third-party products.	20%	15	3
The lack of scientifically relevant sample data impacts Partner A's ability to validate the product.	35%	20	7
There won't be time for Partner A to provide feedback on the format of Analysis reports, which means they could find the reports unacceptable during validation.	10%	5	0.5
Partner A employees are not available to validate the new features until too late in the process, limiting our ability to make additional releases that address any issues they might uncover.	20%	5	1
There won't be time in the QA process to validate, equally, on all browsers on all operating systems.	40%	5	2
Partner A may require more end-user documentation than has been provided.	25%	20	5
Exposure:			18.5

# Kockázatmenedzsment agilis környezetben





# Tartalom

- Konfigurációmenedzsment, verziókezelés, változáskezelés
- Kockázatmenedzsment
- Minőségmenedzsment 
- Mérés és elemzés



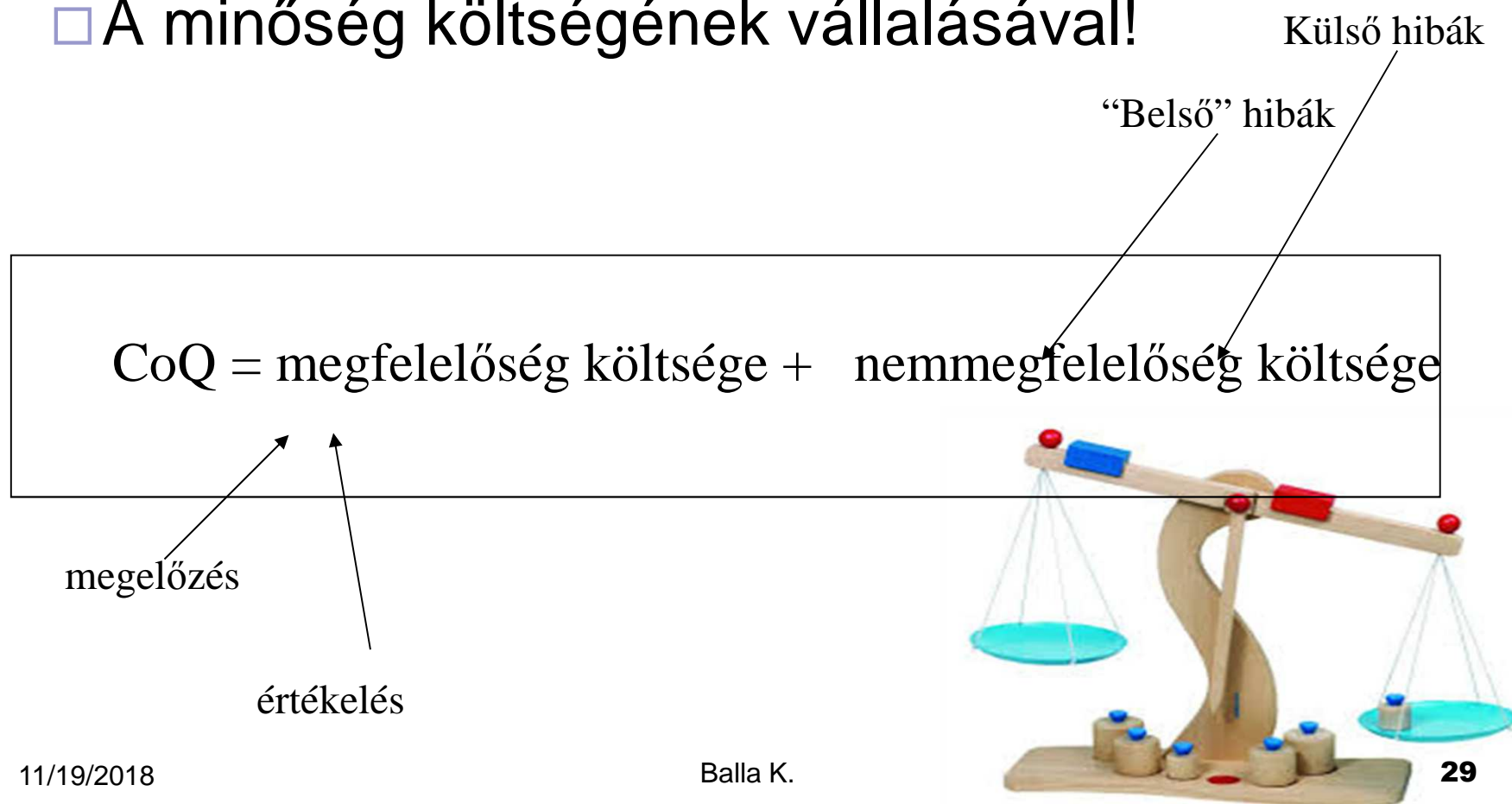
# Minőség, tesztelés és mérés t

- A fontos minőségi attribútumok értékéről  
méréssel kell meggyőződnünk
- A mérések egy részét teszteléssel végezzük
  - ...de vannak jellemzők, melyek értékét a teszteléstől különböző technikákkal mérjük , pl.: eltelt idő, elköltött pénz, felhasznált erőforrások, hibák száma stb. – ezeket a méréseket nem nevezzük „tesztelés”-nek
- De :
  - A szoftver minőségét nem elegendő teszteléssel biztosítani!
  - Új hibaelkerülési, hibamegelőzősi technikák szükségesek a megváltozott szoftverfejlesztési körülmények miatt!

# Hogyan biztosítható a szoftver jó minősége?

- Tudatos és folyamatos befektetéssel

- A minőség költségének vállalásával!





# Mi a szoftverminőség?

- A szoftver jó, ha ...
  - ... időben elkészül
  - ... olcsó
  - ... azt csinálja, amit a felhasználó szeretne
  - ... a nap 24 órájában rendelkezésre áll
  - ... barátságos, könnyen tanulható
  - ... maximum 5 sec alatt mindig válaszol
  - ... kódja érthető, könnyen karbantartható
  - ... új környezetben könnyen telepíthető





# A szoftverminőség...

- Nem „egy – és – egyetemes”, nem állandó...
  - Függ a konkrét helyzettől
- **Minőségi profilt** kell meghatározni **minden esetben!**
  - A minőségi profil kialakításakor ismerni kell a szoftverminőség **fontos elemeit** és a **létező megközelítéseket**
  - A cég **konkrét igényeinek** megfelelő elemeket és megközelítéseket kell **kiválasztani**

# A szoftverminőség fogalmának időbeli változása

- A "*számítástechnika hőskorában*":
  - A program jó, ha bizonyos idő után egyszer lefut és a várthoz hasonló eredményeket ad.
- Kb. 1960-1975 : "*mikrohatékonyság*"
  - A program jó, ha alkalmazása egy adott hardver-szoftver környezetben olcsó, az adott kapacitást a konkrét feladat megoldására optimálisan használja ki.
- Kb. 1980-tól : "*makrohatékonyság*"
  - A jó program hordozható, forráskódja több ember számára is érthető, követhető és a legmesszebbmenőkig felhasználóbarát.
- '90-es évektől:
  - Programrendszerek helyessége a kérdés
  - Módszertanok, Case-eszközök , TQM, "Beépített intelligencia"





# Garvin szoftverminőség definíciói

- **Transzcendens**
  - „Nem tudom meghatározni, de felismerem”. A minőség a veleszületett kiválóságot jelenti.
- **Termék alapú**
  - A minőség precíz és mérhető változó. A minőségi különbségek a termékek egyes összetevőinek vagy jellemzőinek a különbségeiből fakadnak. A minőség tehát nem megítélés kérdése, hanem a termékekben rejlő - objektív - jellemző.
- **Felhasználói alapú**
  - A minőség a felhasználásra való alkalmasság.
- **Érték alapú definíció**
  - A minőséget a költség függvényében határozza meg. Eszerint a jó minőségű termék alacsony áron alkalmas a kitűzött feladat elvégzésére, illetve elfogadható nagyságú költségek mellett felel meg a specifikációjának.
- **Folyamat alapú**
  - A szoftver jó, ha a fejlesztési folyamata jó.



# A minőség mint üzleti kategória

A szoftverfejlesztő cégeknek **bizonyíthatóan** jó minőségű szoftvert kell előállítaniuk; különben nem tudnak a piacon maradni.



# A szoftver auditálása

- Auditálási módszereket definiáltak és alkalmaznak.
- Beszéljünk röviden ezekről:
  - Az auditálás alapelvei
  - Auditok elemei
    - Audit folyamat, audit csapat, audit dokumentáció
  - A szoftvercégeknél végzett auditok sajátosságai
  - Auditorok viselkedési kódexe
  - Szoftvertermékek auditálása – szoftverfolyamatok auditálása



# Az auditálás alapelvei

- Az audit egy **szisztematikus, bizonyítékokat gyűjtő folyamat**.
  - Nem hibavadászat!
  - Nem saját vélemény megfogalmazása!
  - Nem konzultáció / tanácsadás!
  - Nem ítélet cégekről, folyamatokról vagy emberekről!
- Az auditok legyenek **függetlenek**, a bizonyítékokat **objektív módon** kell értékelni, hogy megértsük, mennyire felelnek meg az **audit kritériumoknak**
- Auditáló szervezetekre vonatkozó szabványok, kapcsolódó testületek:
  - ISO 19011:2011 : Guidelines for auditing management systems
  - <http://www.ifac.org/auditing-assurance/clarity-center/clarified-standards>
  - 36 nemzetközi szabvány van (International Standards on Auditing ISAs - and International Standard on Quality Control - ISQC)
  - Nemzeti testületek
    - Auditoroknak, auditorokat akkreditáló szervezetek

# Egy audit elemei

- Audit folyamat
  - Audit terv
  - **Mintavételezés**
  - Audit bizonyítékok
  - Audit eredményei
    - Erősségek
    - Nemmegfelelőségek
    - Fejlesztési területek
    - ...
- Audit csapat
  - Auditor(ok)
  - Auditáltak
- Audit dokumentáció





# Szoftverfejlesztő cégeknél végzett auditok sajátosságai

- Szoftverfejlesztésről szól
- Kapcsolódhat a szoftver termékhez vagy / és szoftver folyamatokhoz
  - Engineering folyamatok, menedzsment folyamatok, támogató folyamatok, szervezeti folyamatok ...
- **Mintavételes** – a minta legyen reprezentatív!
- Szoftverfejlesztéssel kapcsolatos tudást igényel!
  - Elvek, módszerek, eszközök a szoftverfejlesztésben
  - Szükséges a szoftverfejlesztői tapasztalat!

# Auditorok viselkedési kódexe





# 3 auditálási módszer



Evaluating software products  
According to ISO 25000



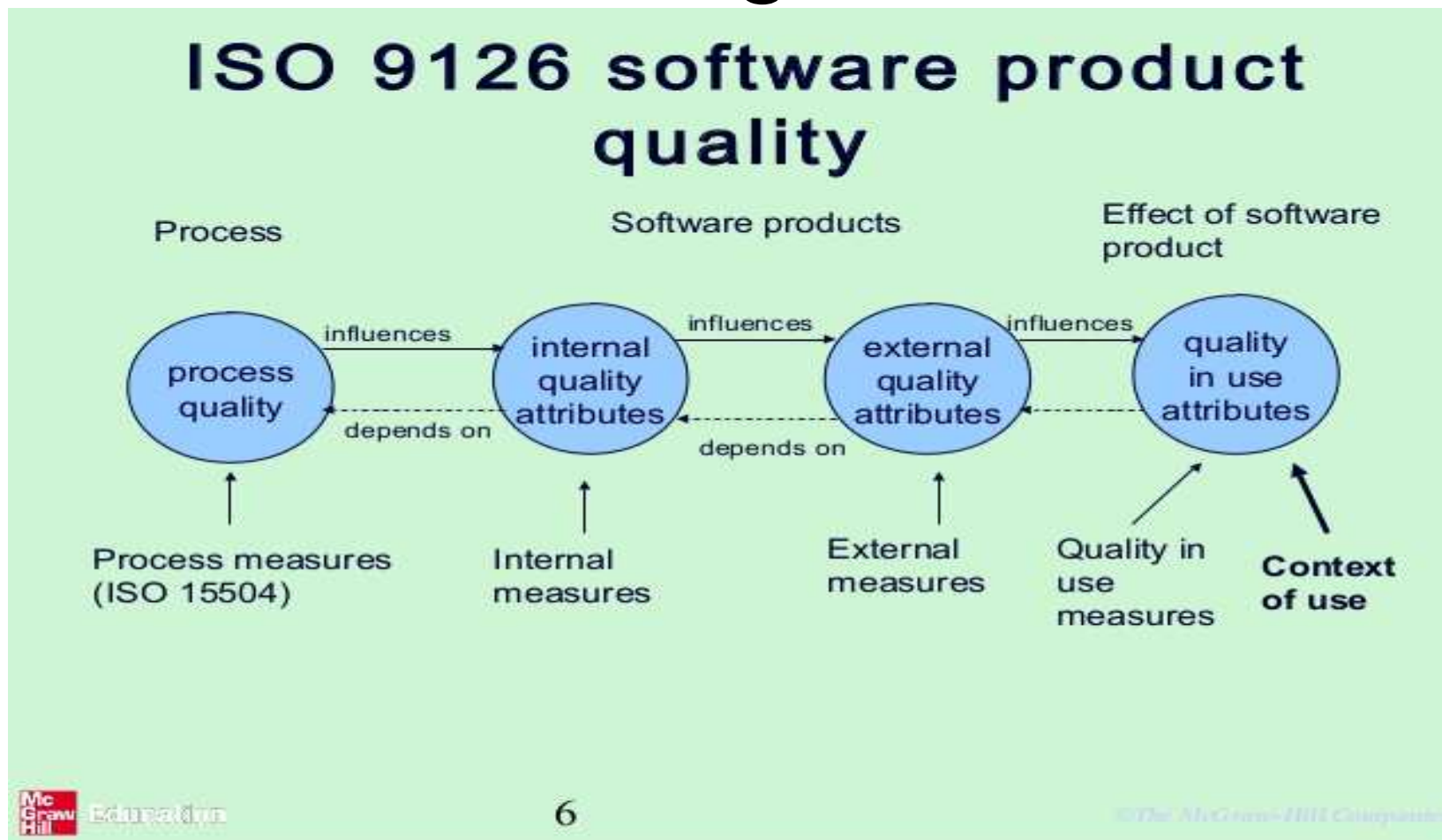




# ISO 25000 szabványcsalád

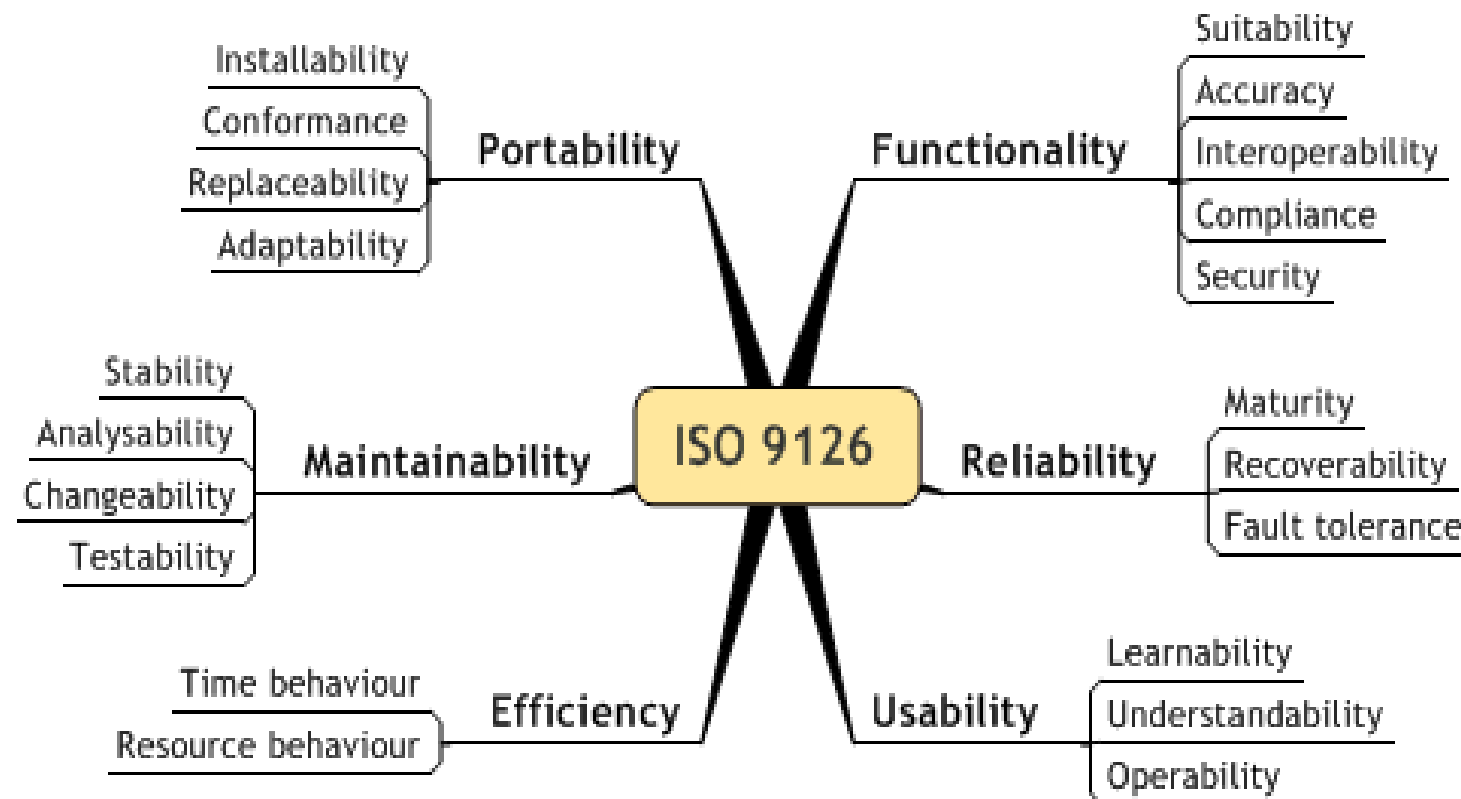
- Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models. Released in 2014.
- Felhasználja a korábbi szabványokat:
  - ISO 9126 termék minőségi attribútumokat és a hozzájuk kapcsolódó metrikákat írja le
  - „Software Product Evaluation: Quality Characteristics and Guidelines for their Use”.
    - Ezt a modellt a 4. előadásban ismertettük, a nem-funkcionális / minőségi követelményeknél
  - ISO 14598 : a vizsgálat módjához ad útmutatást.

# ISO 9016 minőségi modell



©The McGraw-Hill Companies, ISO 9126 software product quality  
<https://image.slidesharecdn.com/softwareproductquality-160127123752/95/software-product-quality-6-638.jpg?cb=1453898412>.

# ISO 9126



Source: <https://www.johner-institut.de/>



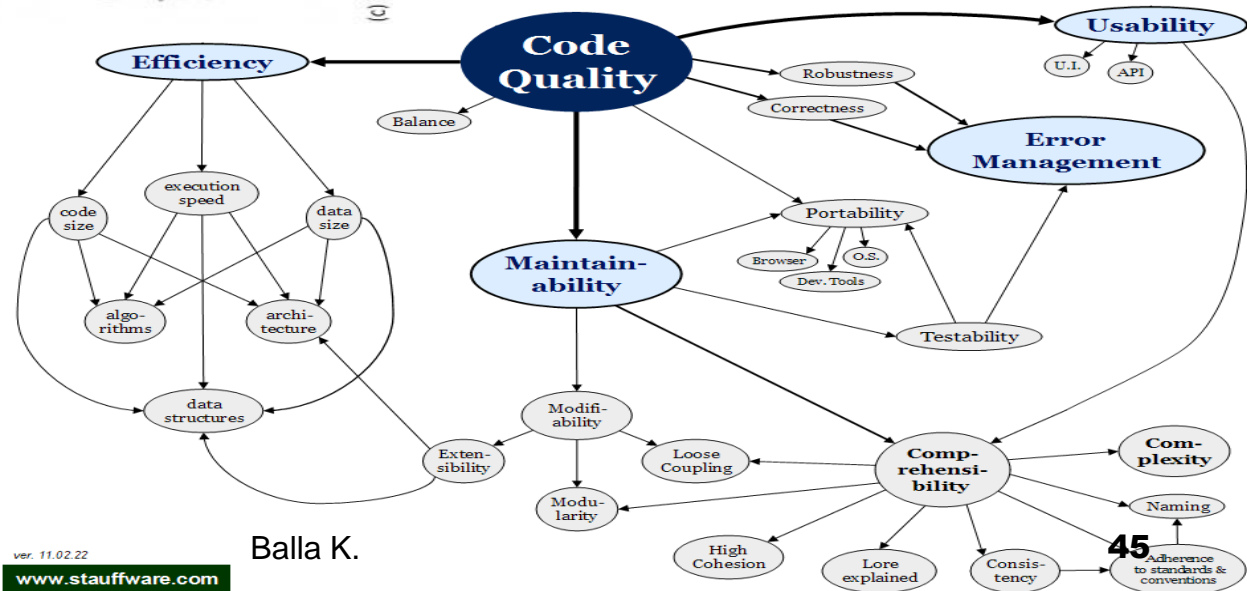
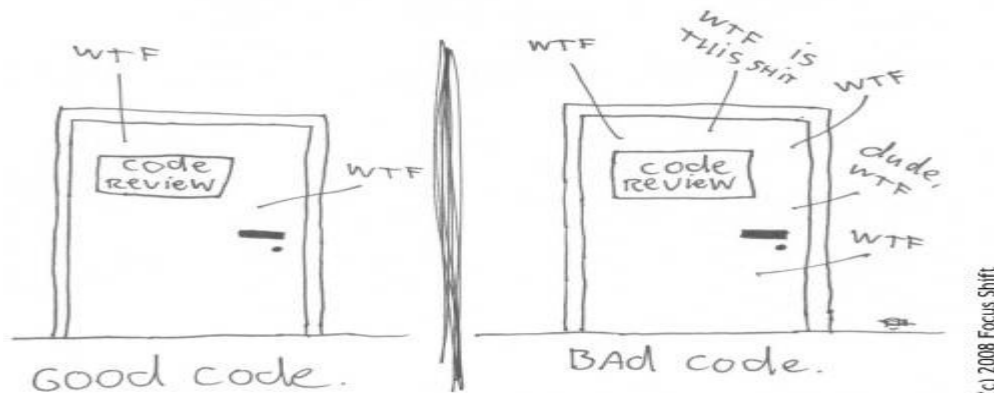
# Szoftver **termék** minőségi követelmények

Vonatkozhatnak:

- Kódra + dokumentumokra + adatokra +...
  - A dokumentumok kapcsolódhatnak:
    - Specifikációhoz
    - Tervezéshez
    - Implementáláshoz
    - Teszteléshez
    - ...

# Különböző megközelítések... Kódminőség

The ONLY valid measurement  
of code quality: WTFs/minute





# Kódminőség

## ■ Eszközökkel

- ☐ Automatikus kódszemlézés
- ☐ Statikus kódelemzés
- ☐ Dinamikus kódelemzés
- ☐ Open source / zárt
- ☐ Programnyelvek segítségével

☐ ...

☐ <https://www.sonarqube.org/>

- A continuous inspection engine to manage the technical debt: unit tests, complexity, duplication, design, comments, coding standards and potential problems. Supports languages: ABAP, Android (Java), C, C++, CSS, Objective-C, COBOL, C#, Flex, Forms, Groovy, Java, JavaScript, Natural, PHP, PL/SQL, Swift, Visual Basic 6, Web, XML, Python.

☐ <https://www.sonarqube.org/features/clean-code/>



# Ne feledjük!

- 1. A kód minőségének javításához erőfeszítést kell tenni!
  - ☐ Szemlézzük a kódot
  - ☐ Írjunk kódolási szabványokat és használjuk őket
  - ☐ **Gyűjtsünk információt** a kód minőségéről, hogy ellenőrizni tudjuk a fejlődést!
- 2. A kódon sok minden mérhető.
  - ☐ De előbb gondolkodjunk!
    - Megéri megmérni?
    - Tényleg értjük, amit mérünk?
    - Segíteni fog a mérés jobb kódot írni?




# Ne feledjük!

A szoftver termék minősége

**sokkal több** a kód minőségénél!!!

- Egyeztetések , kockázatmérséklési tevékenységek sorozata!
- A jó termékhez fegyelmezett munkavégzés szükséges!
  - ☐ Definiáljuk a folyamatokat
  - ☐ Használjuk őket
  - ☐ ... akkor is, ha épp pánikba estünk!





# Szoftverminőségi jellemzők ellenőrzése

- Hasonlít a szoftverfejlesztés folyamatához
  - ☐ Minőségi követelmények specifikációja
  - ☐ A vizsgálati mód meghatározása
  - ☐ A vizsgálat megtervezése
  - ☐ A vizsgálat elvégzése
  - ☐ Visszajelzés a szervezetnek

# ISO 9126-4 – termékminőség mérési modell

	Activity 1	Activity 2	Activity 3	Activity 4	Activity 5	Activity 6	Activity 7	Activity 8
Phase	Requirement analysis (Software and systems)	Architectural design (Software and systems)	Software detailed design	Software coding and testing	Software integration and software qualification testing	System integration and system qualification testing	Software installation	Software acceptance support
9126 series model reference	Required user quality, Required internal quality, Required external quality	Predicted quality in use, Predicted external quality, Measured internal quality	Predicted quality in use, Predicted external quality, Measured internal quality	Predicted quality in use, Measured external quality, Predicted external quality, Measured internal quality	Predicted quality in use, Measured external quality, Predicted external quality, Measured internal quality	Predicted quality in use, Measured external quality, Measured internal quality	Predicted quality in use, Measured external quality, Measured internal quality	Measured quality in use, Measured external quality, Measured internal quality
Key deliverables of activity	User quality requirements (specified), External quality requirements (specified), Internal quality requirements (specified)	Architecture design of Software / system	Software detailed design	Software code, Test results	Software product, Test results	Integrated system, Test results	Installed system	Delivered software product
Metrics used to measure	Internal metrics (External metrics may be applied to validate specifications)	Internal metrics	Internal metrics	Internal metrics External metrics	Internal metrics External metrics	Internal metrics External metrics	Internal metrics External metrics	Quality in use metrics Internal metrics External metrics

11/19/2018

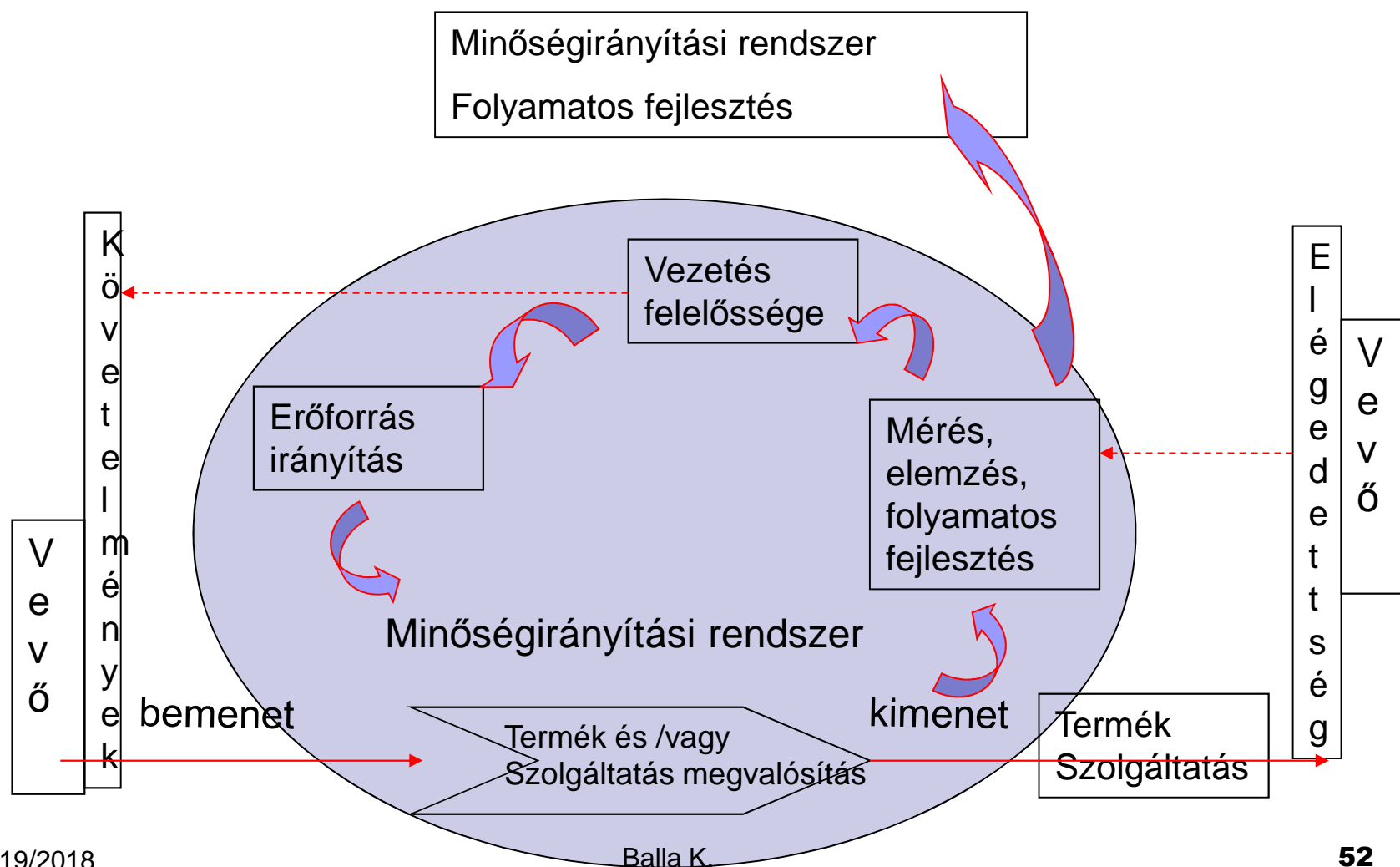
Balla K.



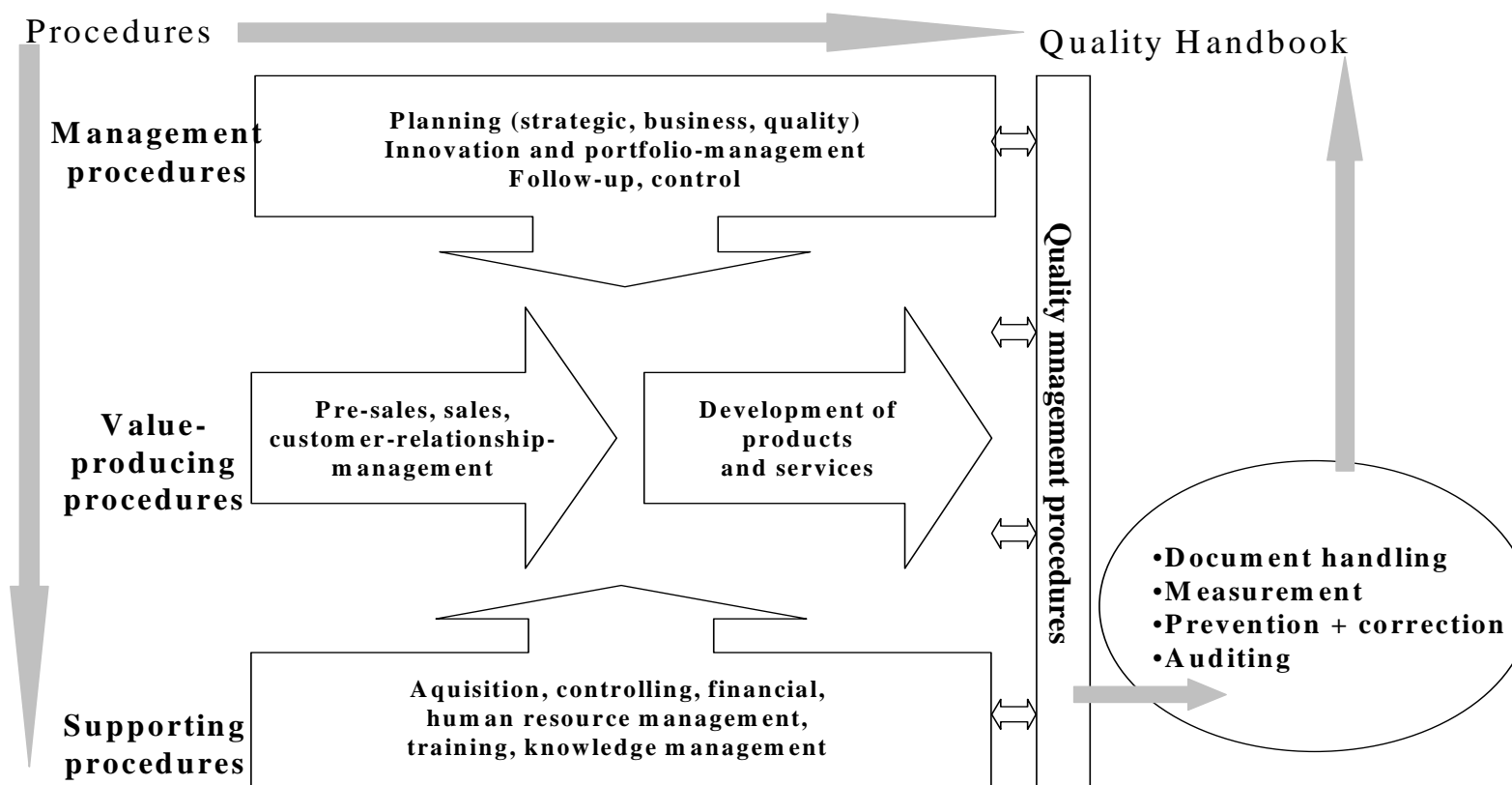
# Folyamat alapú megközelítés

- ISO 9001:2000 / 2008 / 2015
- Minőségirányítási rendszerek. Követelmények
- <https://www.iso.org/>

# Minőségirányítási folyamat modell: ISO 9001:2000



# Példa egy szoftvercég minőségügyi rendszerére





# Az ISO 9001előnyei és hátrányai





# CMMI auditok

- SCAMPI módszer
- <https://sas.cmmiinstitute.com/pars/>
- <http://partners.cmmiinstitute.com/cmmi-appraisals/process-maturity-profiles/>



# Auditok eszközei


- Mintavétel
  - Reprezentatív
- Interjúk
- Dokumentumok ellenőrzése



# Példa: mintavétel SCAMPI-ban

- Basic unit, Support function
- Subgroups

$$\text{Minimum number of Basic Units to be selected from a given Subgroup} = \frac{\text{Number of Subgroups} \times \text{Number of Basic Units in the given Subgroup}}{\text{Total number of Basic Units}}$$




# PPQA- Folyamat és termék minősegbiztosítás a CMMI-ben

- Célja: objektív betekintés biztosítása folyamatokra és a kapcsolódó munkatermékekre vonatkozóan egyaránt. Visszacsatolás biztosítása a projektcsapatnak és menedzsereknek.
- SG 1 Folyamatok és munkatermékek objektív értékelése
  - SP 1.1 Folyamatok objektív értékelése
  - SP 1.2 Munkatermékek és szolgáltatások objektív értékelése
- SG 2 Objektív betekintés nyújtása
  - SP 2.1 Nemmegfelelőségi ügyek kommunikálása és megoldásuk
  - SP 2.2 Feljegyzések készítése



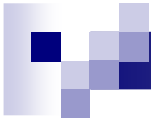
# Tartalom

- Konfigurációmenedzsment, verziókezelés, változáskezelés
- Kockázatmenedzsment
- Minőségmenedzsment
- Mérés és elemzés 



# Mérés a szoftverfejlesztésben

- Fontos összetevő annak biztosításában, hogy tudjuk, hol tartunk és mire számíthatunk – vagyis a teljes szoftverfejlesztés ellenőrzésének eszköze
  - Sok vonatkozásban (kódminőség, hibák, ráfordítás, költségek...)
- A méréseket különböző folyamatok támogatják
  - Tesztelés
  - Minőségbiztosítás
  - Konfigurációmenedzsment
  - Projektmenedzsment
  - Mérés és elemzés...



# Mérés

- ... az a folyamat, amelynek során a valós világ elemeinek attribútumaihoz számokat vagy szimbólumokat rendelünk, abból a célból, hogy valamilyen, jól meghatározott szabály alapján leírjuk, jellemezzük őket. (Fenton)



# Miért mérünk?

- Valamilyen céllal.
- „Nem tudjuk irányítani azt, amit nem tudunk megmérni.” (Tom DeMarco)
- Azért mérünk,
  - hogy dolgokat, összefüggéseket megértsünk
  - hogy irányítani tudjuk az eseményeket
  - hogy az elkészült dolgokat, elvégzett feladatokat értékelni tudjuk, össze tudjuk hasonlítani egymással
  - hogy a jövőben bekövetkező eseményeket minél pontosabban „meg tudjuk jósolni”



# Eredményes méréshez...

- Érteni kell, hogy milyen elem (objektum) milyen vonatkozását (attribútumát) mérjük, és miért
- A sok lehetőség közül ki kell tudnunk választani az adott célnak megfelelőt
- Jó mérőszámot és mérési módszert kell választani
- A mérés eredményét elemezni kell és fel kell használni



# A „jó mérőszámok” kritériumai

## ■ Watts, 1987:

- Objektivitás: Az eredménynek függetlennek kell lennie minden szubjektivitástól. Nem számíthat, hogy ki végzi a mérést.
- Megbízhatóság: Az eredmény legyen pontos és megismételhető.
- Érvényesség: A mérőszám a helyes jellemzőt mérje.
- Szabványosság: A mérőszám legyen egyértelmű és tegye lehetővé az összehasonlítást.
- Összehasonlíthatóság: A mérőszám legyen összehasonlítható más, azonos típusú mérőszámokkal.
- Gazdaságosság: Minél egyszerűbb és olcsóbb egy mérőszám alkalmazása, annál jobb.
- Hasznosság: A mérőszámnak egy igényhez kell kapcsolódnia, nem lehet „önmagáért való”.





# Direkt és indirekt mérés

## ■ Direkt mérés

- egy attribútum vagy entitás mérése nem feltételezi más attribútumok vagy entitások bevonását. (pl. hosszúság)

## ■ Indirekt mérés

- egy attribútum vagy entitás mérése csak további entitások vagy attribútumok bevonásával lehetséges (pl. sűrűség = tömeg / térfogat)



# A mérések eredménye

- Igyekszünk a megfigyelt rendszer elemeit egy számokkal jelölt rendszerbe leképezni
- Cél, hogy a numerikus rendszerben az attribútumok milyenségét jelölő számértékekkel valamilyen műveleteket végezzünk, s ebből megértsük a „milyenségüket”
- A leképezéshez használt rendszer megszabja, hogy milyen típusú elemzést végezhetünk a mért értékekkel
- A különbségek miatt bevezették a mérési skálákat.
  - Nominal, ordinal, interval, ratio, absolute



# Objektív és szubjektív mérőszámok

- Objektivitásra törekszünk
- A szubjektív mérőszámok is jók lehetnek
  - ha megértjük a szubjektív mérőszám korlátait
  - Pl: egy specifikáció minőségének meghatározása
    - 1: rossz, 5: jó
    - Ha pl. az interfészre vonatkozó követelmények között sok „rossz” szerepel, ezt felül kell vizsgálni



# Szubjektív mérőszámok a szoftver esetében

- Fontos és jó, de nem biztos, hogy szigorú értelemben vett mérés
- Példák: Likert skála:
  - Pl: A program megbízható (Teljesen egyetértek, egyetértek, nem tudom, nem értek egyet, egyáltalán nem értek egyet)
  - „Erőltetett” besorolás: n alternatíva, 1 (legrosszabb)-tól n (legjobb)-ig
    - Pl: soroljuk be a következő szv modulokat a karbantarthatóság nehézsége szerint. 1 : legkevésbé nehéz, 5 : legnehezebb
  - Gyakorisági skála
    - Pl: milyen gyakran fagy le a rendszer? (állandóan, gyakran, néha, ritkán, soha)
  - Rendezési skála
    - Pl: milyen gyakran fagy le a rendszer? (1: óránként, 2: naponta, 3: hetente, 4: havonta, 5: évente néhányszor, 6: évente egyszer v. kétszer, soha)
  - Összehasonlítás
    - Pl: A és B: 1: sokkal jobb...5 : nagyjából azonos....10: sokkal rosszabb
  - Numerikus skála
    - Pl: 1: nem fontos.....10: fontos



# Szoftver mérések

- A szoftver mérésének őskora, „primitív” mérőszámok
  - Utasításstatisztikák
  - Hibastatisztikák
- Programok irányított gráf reprezentációja
- Ciklomatikus szám
- Hívási komplexitás
- .....további modellek....

$M = E - N + 2P$ ,  
where  
 $E$  = the number of  
edges of the graph.  
 $N$  = the number of  
nodes of the graph.  
 $P$  = the number of  
connected  
components.




# Tipikus szoftver-mérőszámok

## ■ Direkt mérőszámok (példák):

- ☐ forráskód hossza
- ☐ tesztelési folyamat időtartama
- ☐ a tesztelés során megtalált hibák száma
- ☐ egy programozó által a projekten töltött idő

## ■ Indirekt mérőszámok (példák):

- ☐ programozó teljesítménye: megírt LOC / ember-hónap
- ☐ hibasűrűség egy modulban: hibák száma / modul mérete
- ☐ hibamegtalálási hatékonyság: megtalált hibák száma / összes hibák száma
- ☐ követelmények stabilitása: kezdeti követelmények száma / összes köv. száma



# Termék minőségi attribútumok mérése az ISO 9126 szabványban

- Részletes útmutatók a belső, külső és használat közbeni minőség mérésére

# Használat során tapasztalt minőség metrikák.

## Alkalmassági (suitability) mérőszámok (példa)


Metric Name	Purpose of the metrics	Method of application	Measurement, formula and data element computations	Measurement scale	Scale type	Measure type	Input to measurement	Perspective Beneficer
<b>Task effectiveness</b>			<b>task effectiveness <math>M1 = A \times B</math></b>  A= Quantity (completeness) = the proportion of task goals represented in the output of task B= Quality (correctness) = the degree to which the task goals represented in the output have been achieved	$0 \leq M1 \leq 1$  The closer to 1.0 is the better.	M1= Abs. A= Abs. B= Abs.	A=Count B=Count  M1= Count x Count		
<b>Task productivity</b>			<b>task productivity <math>M2 = M1 / T</math></b>  A = task effectiveness T = task time	$0 \leq M2$  The larger is the better.		M2= Ratio  M2= (Count x Count)/ Time		
<b>Productive proportion</b>			<b>productive proportion <math>M3 = Ta / Tb</math></b>  Ta = productive time = task time - help time - error time - search time Tb = task time	$0 \leq M3 \leq 1$  The closer to 1.0 is the better.	M3 = Abs.	Ta=Time Tb=Time  M3= Time/ Time		
<b>Relative user productivity</b>			<b>Relative user productivity <math>M4 = A / B</math></b>  A = ordinary user's task productivity = M2 B = expert user's task productivity = M2	$0 \leq M4 \leq 1$  The closer to 1.0 is the better.	M3 = Abs.	M4= M2 / M2		
<b>Attitude</b>			<b>SUMI scale <math>X = A</math></b> A = questionnaire producing psychometric scales	population average is 50 for each scale	Ord.	A= Count X= Count		





# Funkciópont számolás

- A szoftvertermék komplexitásának mérésére és becslésére
- Lásd 5. előadás: Design
  - Módszerek:
    - IFPUG
    - MkII
    - COSMIC



# A projektmenedzsment folyamatok és erőforrások mérése

- Projekthez kapcsolódó mérőszámok, projektkövetés során mérhetők
  - Belső folyamat-attribútumok: a projektre fordított idő, a szükséges erőforrások és a költségek.
    - Ezek mérése egyszerű feljegyzést jelent: fel kell jegyezni egyrészt a tervezett értékeket, másrészt a projekt végrehajtása során megvalósult értékeket.
    - Az idő mérésénél hasznos lehet, ha a tevékenységeket a lehető legjobban alábontjuk, és az egyes résztevékenységekre fordított időt jegyezzük fel.
  - Külső folyamat-attribútumok:
    - pl: ellenőrizhetőség, megfigyelhetőség, stabilitás.



# A projektmenedzsment folyamatok és erőforrások mérése

## ■ Direkt mérések:

- Felhasznált költségek (*Actual Cost of Work Performed - ACWP*)
- Adott időszakra ütemezett munkára tervezett költség (*Budget Cost of Work Scheduled - BCWS*)
- Megvalósult érték = Elvégzett munkára tervezett költség (*Earned Value = Budget Cost of Work Performed - BCWP*)

## ■ Indirekt mérések:


- Elvégzett munka alapján számolt költségeltérés (*Cost Variance - CV = BCWP - ACWP*)
- Elvégzett munka és eltelt idő alapján számolt költségeltérés (*Schedule Variance - SV = BCWP - BCWS*)
- Költség szerinti teljesítmény index (*Cost Performance Index - CPI = BCWP / ACWP*)
- Időzítés szerinti teljesítmény index (*Schedule Performance Index - SPI = BCWP / BCWS*)



# A projektmenedzsment folyamatok és erőforrások mérése

## ■ Az erőforrások mérése

- fontos lehet feljegyezni, hogy szám szerint mennyi és milyen típusú erőforrást használtunk fel.
- Emberi erőforrásoknál jól használható adat lehet az erőforrás szakmai tapasztalata és termelékenységé, mert ezekből következtetni lehet egy bizonyos típusú feladat emberi erőforrás-igényeire.
- Az emberi erőforrások termelékenységének mérése nem egyszerű feladat, mert pl. az adott idő alatt megírt kód nagysága nem biztos, hogy releváns információt szolgáltat.




# A fejlesztési (engineering ) folyamatok mérése

- Idő, költség, erőforrás-felhasználás itt is fontos lehet
- Hibák mérése a különböző folyamatokban (szám, típus/súlyosság, kijavítás ráfordítása)
- A folyamat típusától függően, további attribútumokat lehet meghatározni
  - Tervezésnél: kezdeti követelmények száma, módosult követelmények, új követelmények, törölt követelmények
  - Kódolásnál figyelhető a komplexitás (pl. funkciópont analízissel), valamint a kódolás során felfedezett és kijavított hibák száma és típusa.
  - Tesztelésnél figyelhető a megtalált hibák száma és súlyossága, kijavítás ráfordítása.



# Fenton kerete a szoftvermérésekre

Entitások	Attribútumok	
<i>Termék</i>	Belső attribútum	Külső attribútum
Specifikáció	méret, újrahasználatosság, modularitás, szintaktikai helyesség	mindenre kiterjedő, karbantarthatóság
Tervek	méret, újrahasználatosság, modularitás, összekapcsolódás, összetartozás, funkcionalitás	minőség, komplexitás, karbantarthatóság
Kód	méret, újrahasználatosság, modularitás, összekapcsolódás, funkcionalitás, algoritmikus komplexitás, strukturáltság	megbízhatóság, használhatóság, karbantarthatóság
Teszt adatok	méret, lefedettség	minőség
<i>Folyamat</i>		
Specifikáció készítése	idő, ráfordítás, megváltozott specifikációk száma	minőség, költség-stabilitás
Részletes tervezés	idő, ráfordítás, megtalált specifikációs hibák száma	költség-hatékonyság
Tesztelés	idő, ráfordítás, megtalált hibák száma	költség-hatékonyság, stabilitás
<i>Erőforrás</i>		
Alkalmazottak	életkor, ár	termelékenység, tapasztalat, intelligencia
Csapatok	méret, kommunikáció, strukturáltság	termelékenység, minőség
Szoftver	ár, méret	használhatóság, megbízhatóság
Hardver	ár, gyorsaság, memória mérete	megbízhatóság
Irodák	méret, hőmérséklet, fény	komfortosság, minőség



# Mérés és elemzés (MA) a CMMI-ben

- A mérés és elemzés célja olyan mérési ismerettár (kelléktár) fejlesztése és fenntartása, mely a menedzsment információs igényeit kielégíti.
- SG 1 Mérési és elemzési tevékenységek
  - ☐ SP 1.1 Mérési célkitűzések meghatározása:
  - ☐ SP 1.2 Mérések specifikálása:
  - ☐ SP 1.3 Adatgyűjtési- és tárolási eljárás mód specifikálása:
  - ☐ SP 1.4 Elemzési eljárás mód specifikálás
- SG 2 Mérési eredmények szolgáltatása
  - ☐ SP 2.1 Mérési adatok gyűjtése:
  - ☐ SP 2.2 Mérési adatok elemzése:
  - ☐ SP 2.3 Adatok és eredmények tárolása:
  - ☐ SP 2.4 Eredmények közzététele

# QIP



## ■ Quality Improvement Paradigm

- Basili, 1985
- 2 eszközt használ:
  - GQM (Goal-Questing-Metric paradigm)
  - EF (Experience Factory Organization)

## ■ Software Engineering Laboratory (SEL) – organisation sponsored by National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC)

- <https://sloanreview.mit.edu/article/improve-software-quality-by-reusing-knowledge-and-experience/>
- <https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=12241>

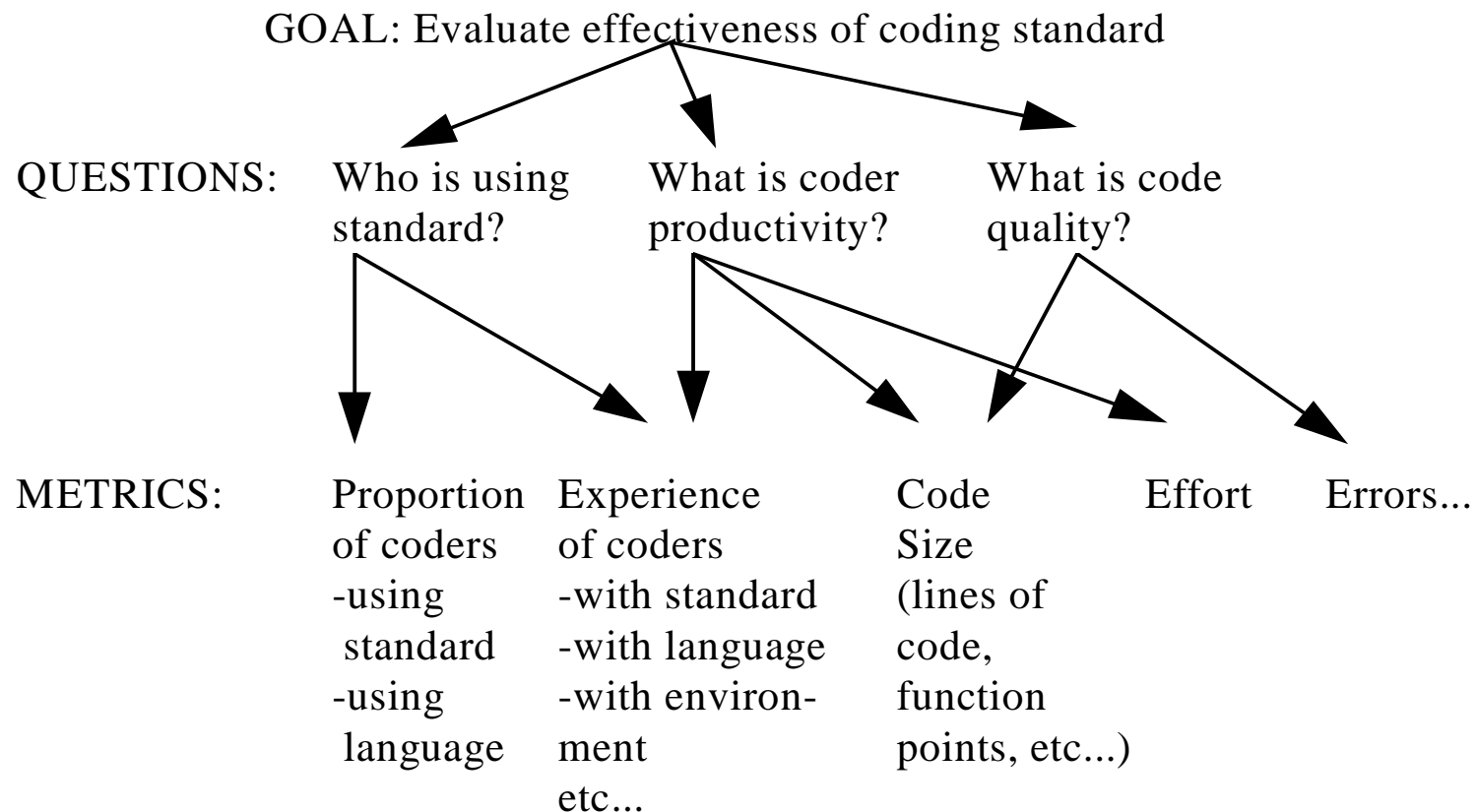


# QIP



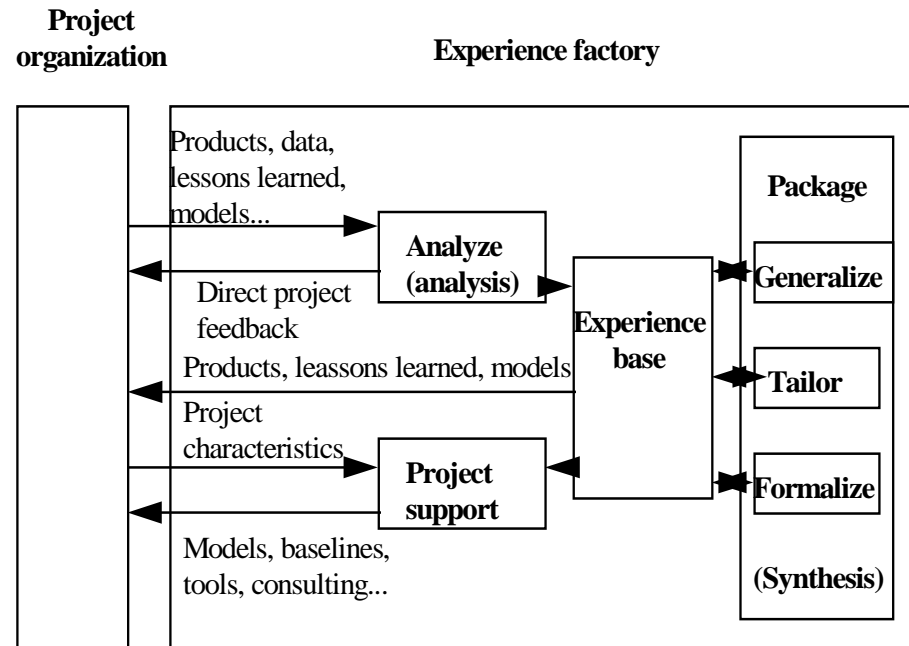
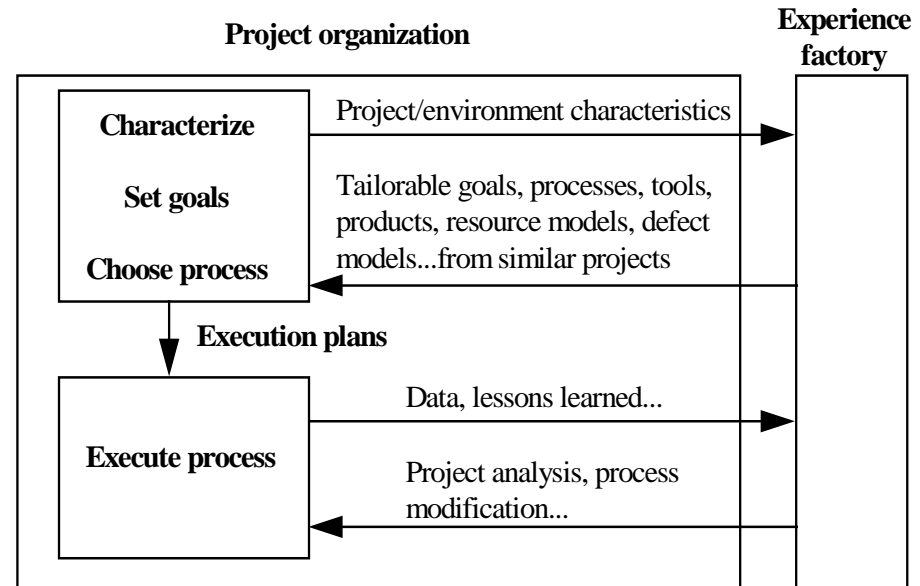
- Folyamatos javításra koncentrál
- Az egyes projektek tapasztalatát elemzi, „csomagolja” további projektekben való felhasználhatóság szempontjából
- Alapfázisai:
  - A szoftverfolyamat megértése
  - Az alkalmazott technológiák (munkamódszerek) hatékonyságának figyelése, mérésekkel. Annak meghatározása, hogy mely technológiák megfelelőek az adott környezetben.
  - A tapasztalatok „csomagolása”. Szabványok, képzés, fejlődési stratégia kialakítása.

# A GQM alkalmazása. Példa



# EF

- Változó szerepek a szervezeten belül





# Mérés agilis környezetben

- Folyamatos tevékenység
- <https://www.atlassian.com/agile/metrics>
  - ☐ Sprint burndown
  - ☐ Epic and Release Burndown
  - ☐ Velocity
  - ☐ Control Chart
  - ☐ Cumulative Flow Diagram

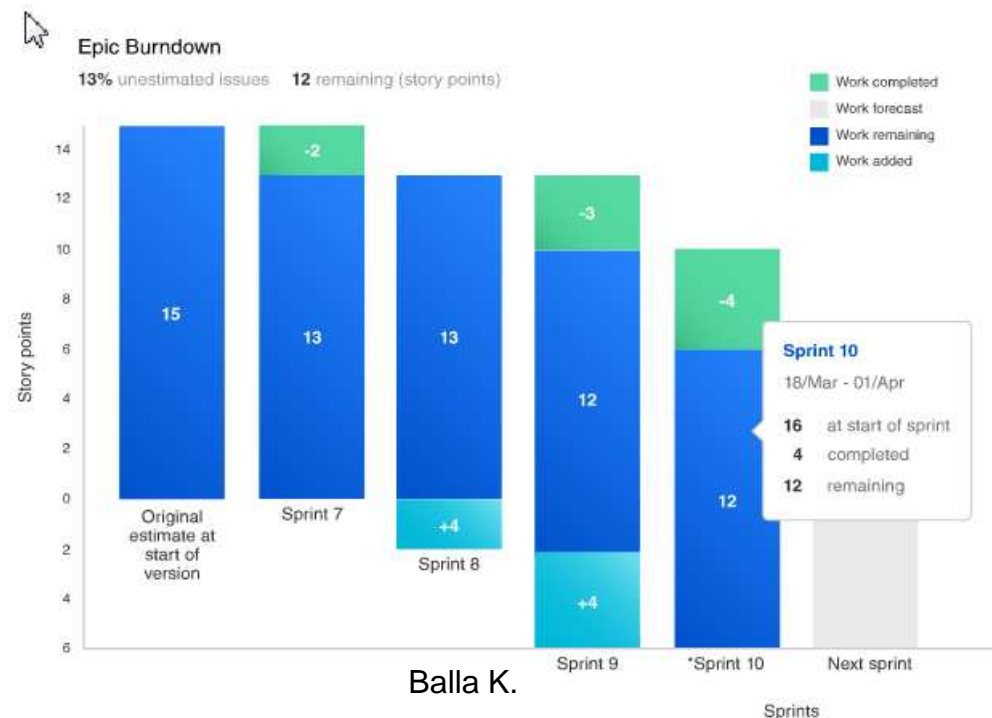
# Burndown chart

- Sprint burndown report tracks the completion of work throughout the sprint. The x-axis represents time, and the y-axis refers to the amount of work left to complete, measured in either story points or hours. The goal is to have all the forecasted work completed by the end of the sprint.



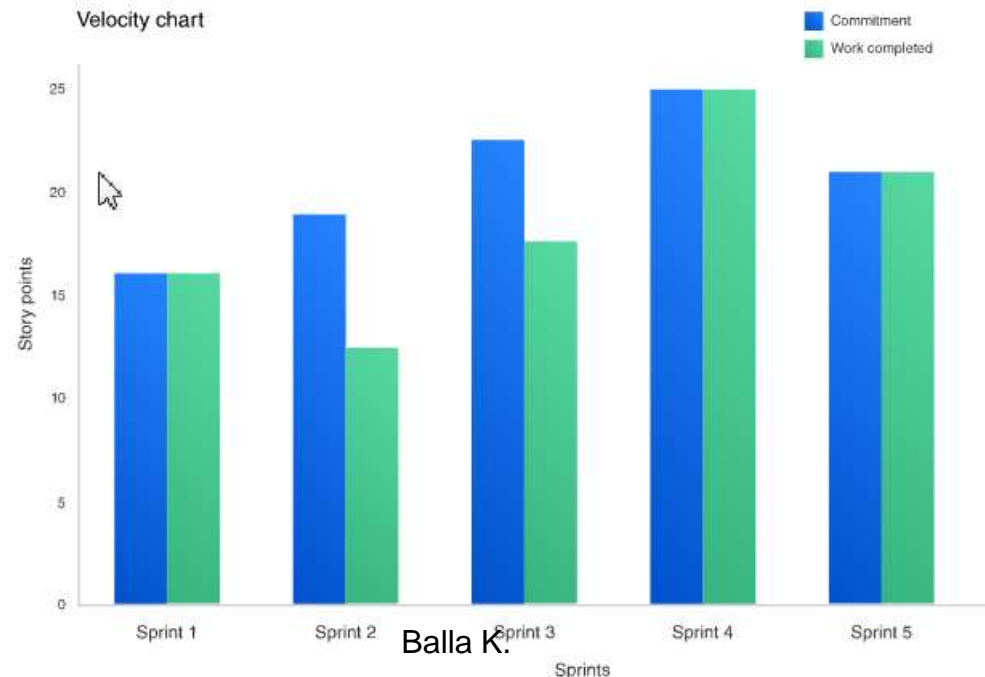
# Epic and release burndown

- Epic and release (or version) burndown charts track the progress of development over a larger body of work than the sprint burndown, and guide development for both scrum and kanban teams. Since a sprint (for scrum teams) may contain work from several epics and versions, it's important to track both the progress of individual sprints as well as epics and versions.



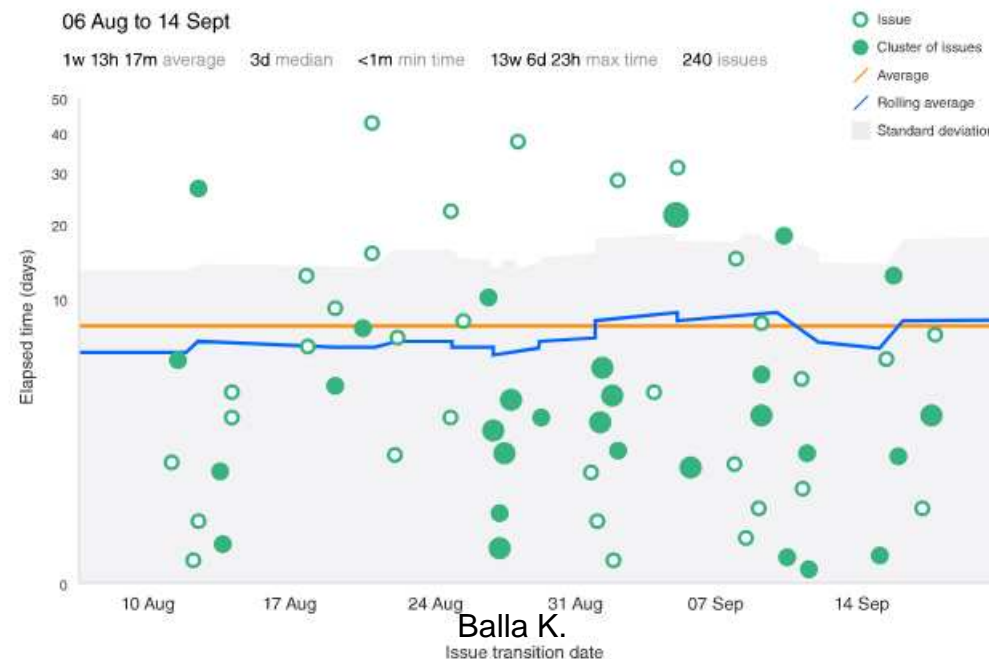
# Velocity

- Velocity is the average amount of work a scrum team completes during a sprint, measured in either story points or hours, and is very useful for forecasting. The product owner can use velocity to predict how quickly a team can work through the backlog, because the report tracks the forecasted and completed work over several iterations—the more iterations, the more accurate the forecast.



# Control charts

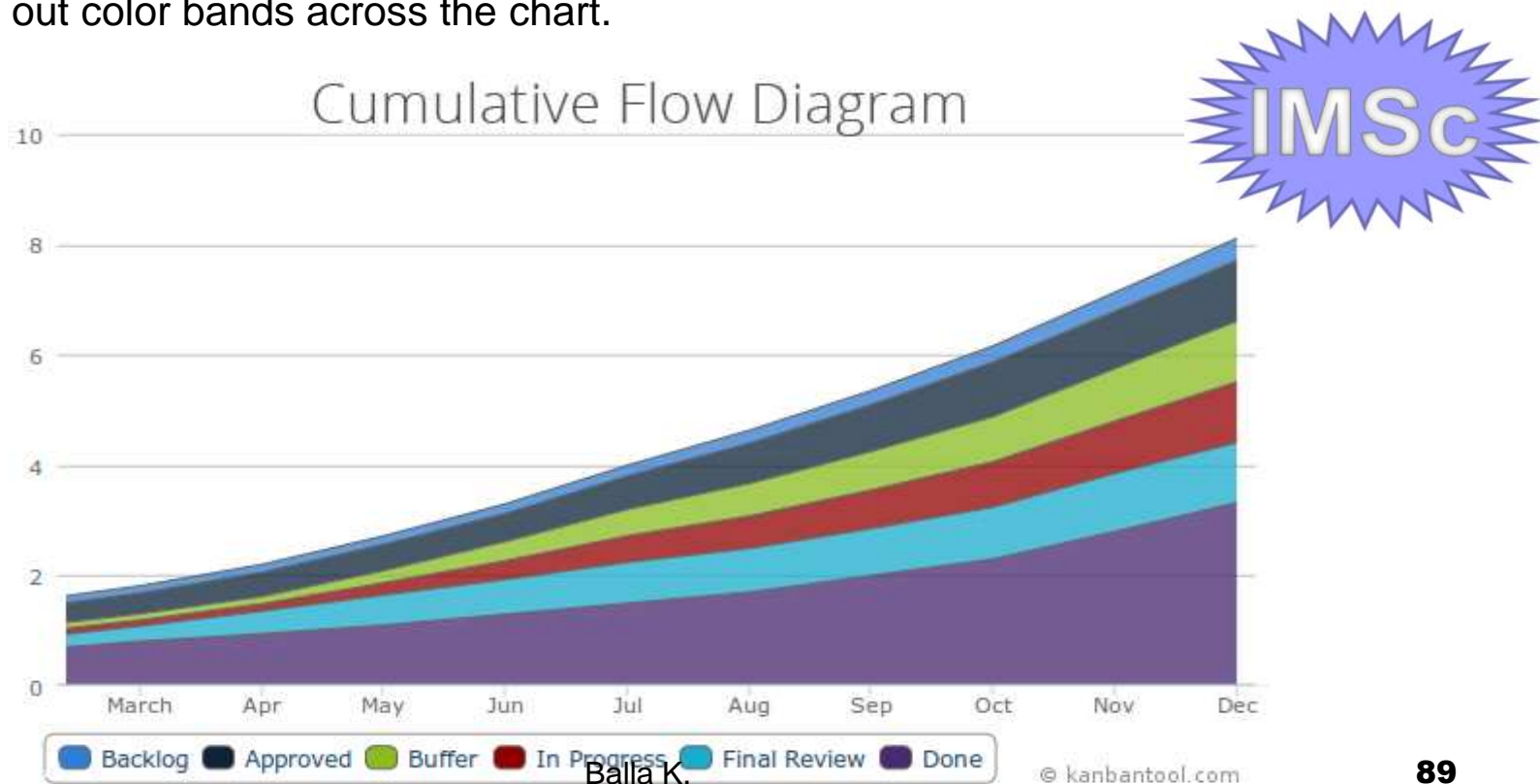
- Control charts focus on the cycle time of individual issues—the total time from "in progress" to "done". Teams with *shorter* cycle times are likely to have higher throughput, and teams with *consistent* cycle times across many issues are more predictable in delivering work.





# The cumulative flow diagram

- The cumulative flow diagram is a key resource for [kanban](#) teams, helping them ensure the flow of work across the team is consistent. With number of issues on the Y axis, time on the X axis, and colors to indicate the various [workflow](#) states, it visually points out shortages and bottlenecks and works in conjunction with [WIP limits](#).
- The cumulative flow diagram should look smooth(ish) from left to right. Bubbles or gaps in any one color indicate shortages and bottlenecks, so when you see one, look for ways to smooth out color bands across the chart.



# Project Cockpit Chart

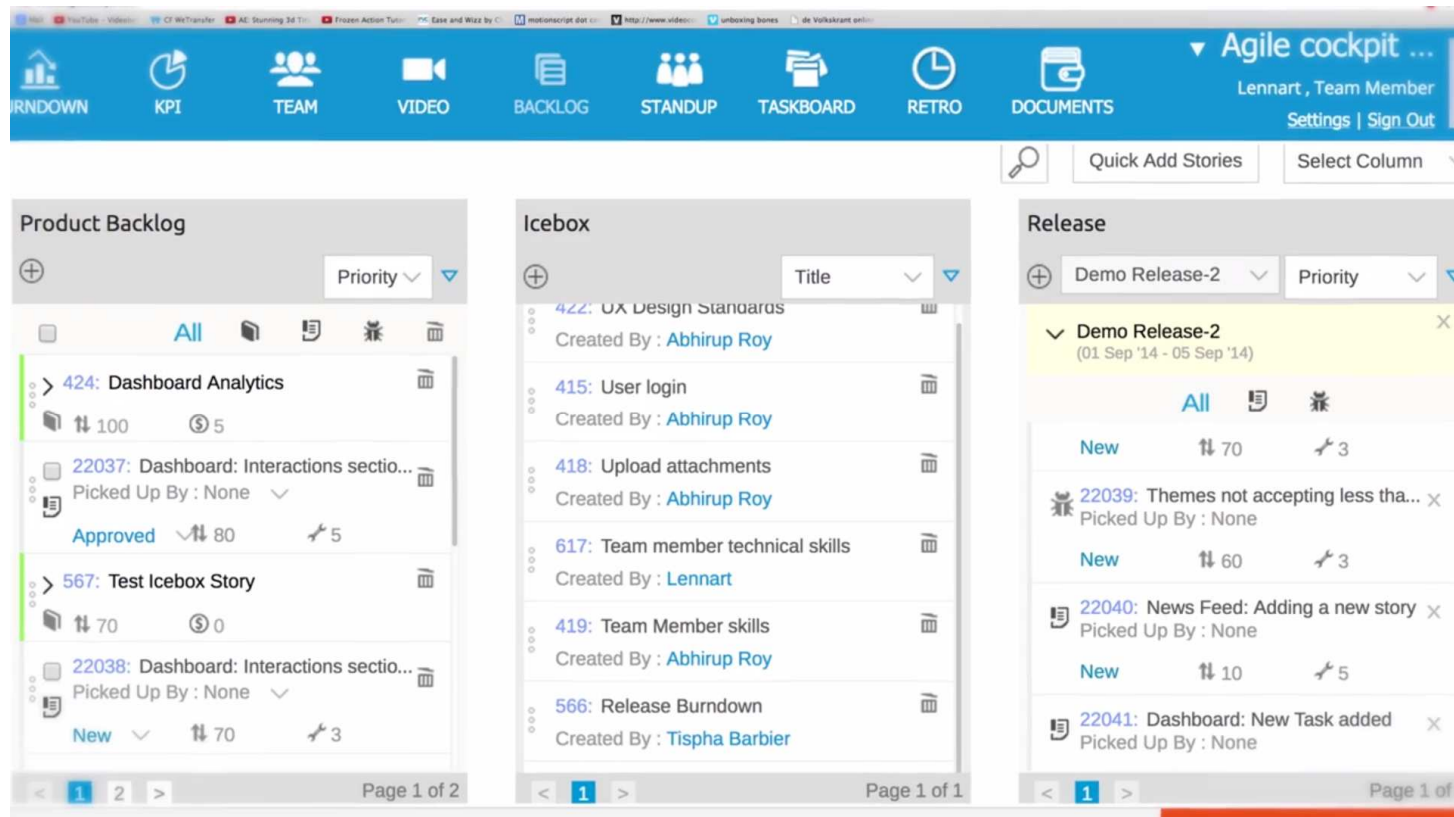
## 26 KPIs for agile offshore Software project

- 9 Business Case (BC)
- 4 Productivity (PR)
- 5 Service Provider Relationship (SPR)
- 8 Quality Assurance (QA)

Example: PR-2: Story Point per PD; PR-5: Velocity



# Project Cockpit Chart



<https://www.agilecockpit.com/media/storing-ideas-in-the-icebox/>

# Agile cockpit



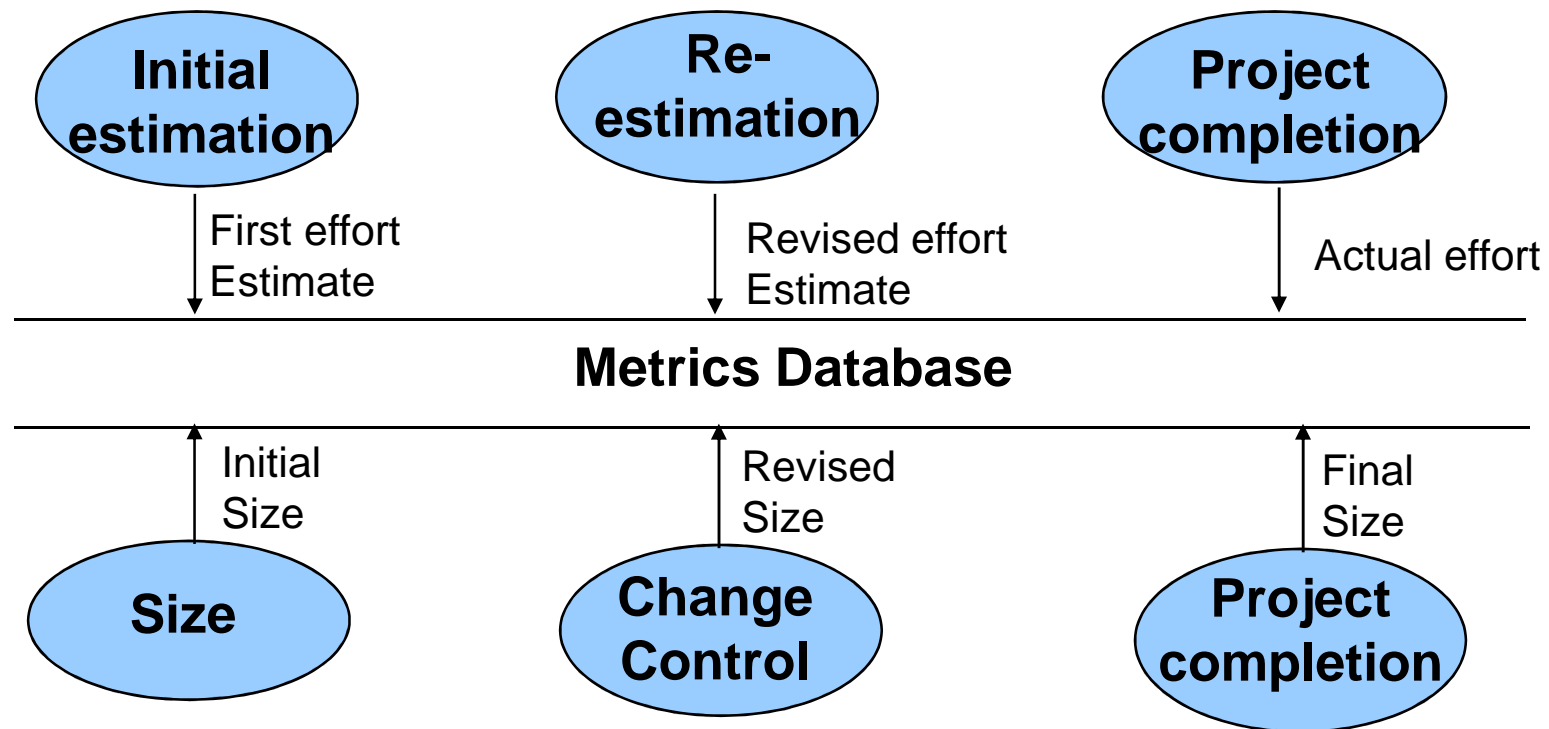
Lufthansa Systems gains competitive edge by standardizing on Atlassian  
<https://www.youtube.com/watch?v=JcOrVDCuXbw>



# Mérési program egy cégnél

- Stratégiai döntés, erős vezetői támogatás
- Szervezet
- Képzés
- Célok (pl. GQM-mel meghatározva)- változhatnak
- Mérési eljárások kialakítása
- Mérési eszközök, technikák
- Mérési adatok adatbázisa
- Folyamatos visszacsatolás, fejlődés

# Mérési adatok tárolása



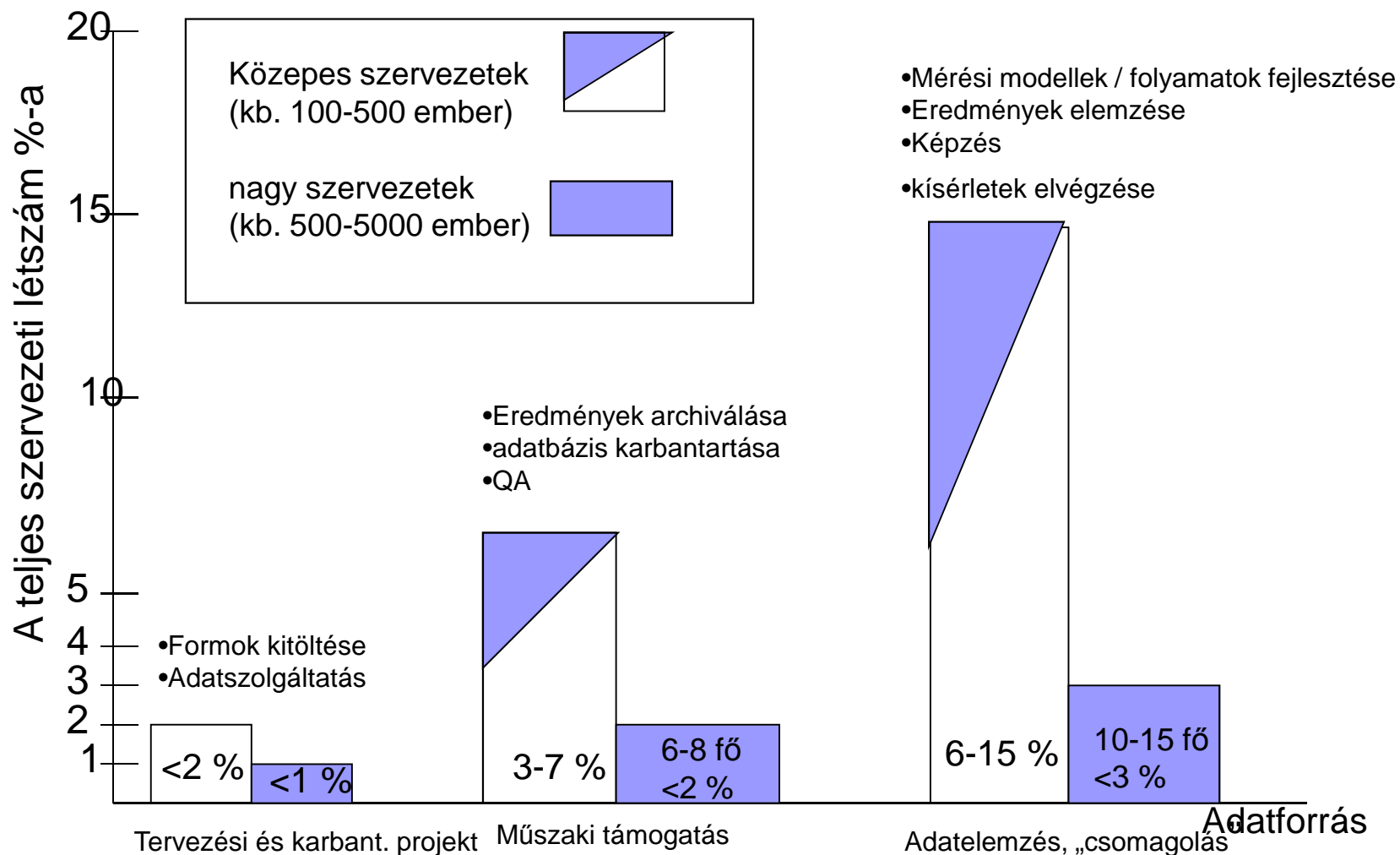


# Adatgyűjtési mechanizmusok

- Formok
- Számítógépes eszközök
- Interjúk

# A szoftvermérés költsége

(Software Measurement Guidebook, NASA, Software Engineering Laboratories, NASA-GB-001-94)







# Miről volt szó...

- Konfigurációmenedzsment, verziókezelés, változáskezelés
- Kockázatmenedzsment
- Minőségmenedzsment
- Mérés és elemzés