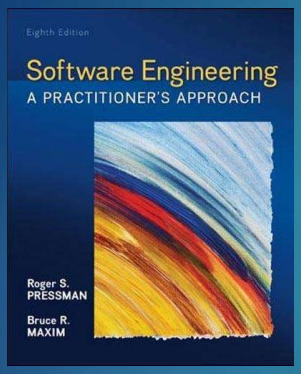
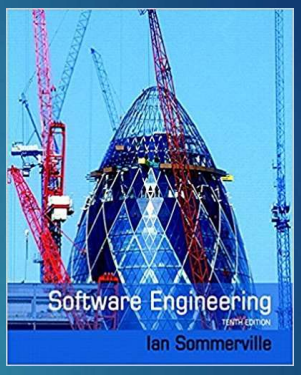
Nagy rendszerek fejlesztésének technológiája

Dr. Tick József

1., Bevezetés



R. S. Pressman, B. R. Maxim:
Software Engineering –
A Practitioer's Approach 8th Ed.
McGraw-Hill, 2015



I. Sommerville:
Software Engineering 10th Ed.
Addison-Wesley, 2015

1.1. A szoftver sajátosságai

3

- A szoftver tekinthető mint szellemi alkotás ↔ értékesíthető termék, illetve szolgáltatás
- A szoftver, mint termék
  - Megjelenik mint önnálló termék vagy mint egy komplex rendszer intelligenciáját növelő elem
  - Speciális elvárások az adott termékre vonatkozóan
    - Funkcionalitás, interoperabilitás → specifikáció
  - Általános elvárások – etikai, technológiai, biztonsági, adat-biztonsági
  - A termékkel szembeni elvárások

A szoftver, mint termék specialitásai

4

- A szoftver termék esetében eltérő a megvalósítás fázisa
  - Nincs „átültetés" az anyagba
  - Nem kell figyelembe venni a tervezés során
    - Az anyagjellemzőket
    - A megmunkálási módok jellemzőit
- Nincs anyag → nincs öregedés, anyag kifáradás, degradáció (pl. kopás, környezeti hatás: UV → műanyag)

A hardver elromolhat, a szoftver soha! Az már akkor rossz, amikor írják! (Mi építjük bele a hibát!)

A hagyományos termék hiba-arány görbéje

5

Az ún. „fürdőkád görbe”

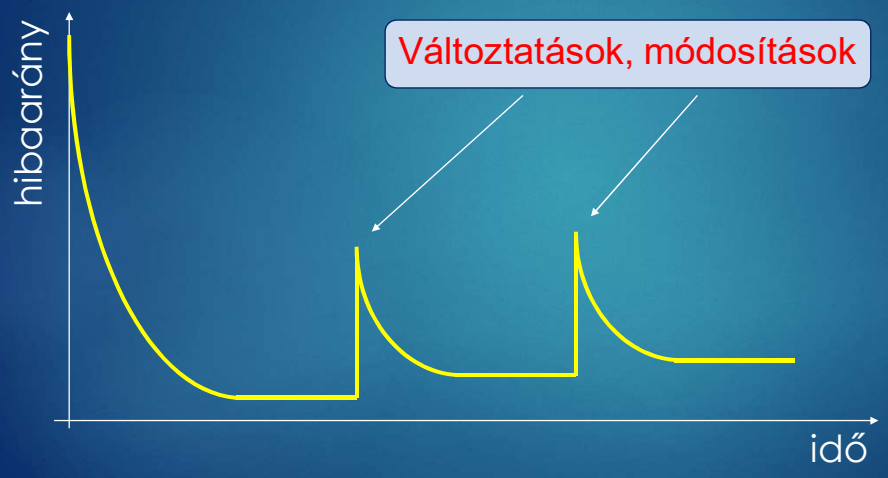
A szoftver termék hiba-arány görbéje

6

Változtatások, módosítások nélkül

A szoftver termék hiba-arány görbéje

7



Változtatások, módosítások miatt folyamatosan romlik a hiba-arány

A szoftver, mint szellemi termék specialitásai

8

- Nincs anyag → könnyen másolható → egyszerűen terjeszthető, de illegálisan eltulajdonítható
- Komoly erőfeszítések az illegális használat megakadályozására
  - Műszaki oldalon – másolás elleni védelem, funkcionális korlátozás, időbeli korlátozás
  - Jogi - szerzői jog védelem

„A szerzői jogról szóló 1999. évi LXXVI. törvény alapján a szerzői jogi védelem az irodalmi, tudományos és művészeti alkotásokat, ún. műveket illeti elsősorban (pl. szépirodalmi művek, zeneművek, filmalkotások, festés, szobrászat útján létrejött alkotások), de védelmet biztosít olyan műtípusok esetében is, mint a **szoftverek, illetve az adatbázisok**." (www.sztnh.gov.hu)

A szoftver fejlesztés jellemzői

9

Szellemi alkotás

- Kreatív, ötletes, egyedi megoldások,
- „míves" kidolgozás

Ipari termék

- Szabványoknak megfelelő mérnöki gyártási folyamat,
- Elveken alapuló fejlesztési módszertanok
- A módszereket támogató eszközök,
- Ipari „sorozatgyártás" – „tömegtermelés"
- Minőségmenedzsment,
- Újrafelhasználás

1.2. A szoftver fejlesztése

10

Szoftver krízis

↓ 1968 NATO konferencia

Software Engineering

↓ Növekvő funkcionalitás, összetett rendszerek, DB-k

Large Scale Software Engineering

↓ Komplex rendszerek (Big Data, Web, mobil, AI

Very / Ultra Large Scale Software Engineering

E. W. Dijkstra (1930-2002)
Holland matematikus, informatikus. Megalkotta a strukturált programozást, bevezette a szemaforok alkalmazását a szálak szinkronizálásához, kitalálta a Dijkstra és a Bankár algoritmusokat.

1.3. A szoftverek alkalmazási területei

(Software Application Domains)

11

- Rendszer szoftverek
- Felhasználói szoftverek
- Mérnöki-tudományos szoftverek
- Beágyazott szoftverek
- Polcos szoftverek
- Web és mobil alkalmazások
- Mesterséges intelligencia szoftverek

Komplex rendszerek több kategóriába is besorolhatók

- nehéz a kategorizálás,
- a besorolás nem mindig egyértelmű,
- sok rendszer esetében különböző típusú elemek azonosíthatók

1.4. A szoftverek természetének változása

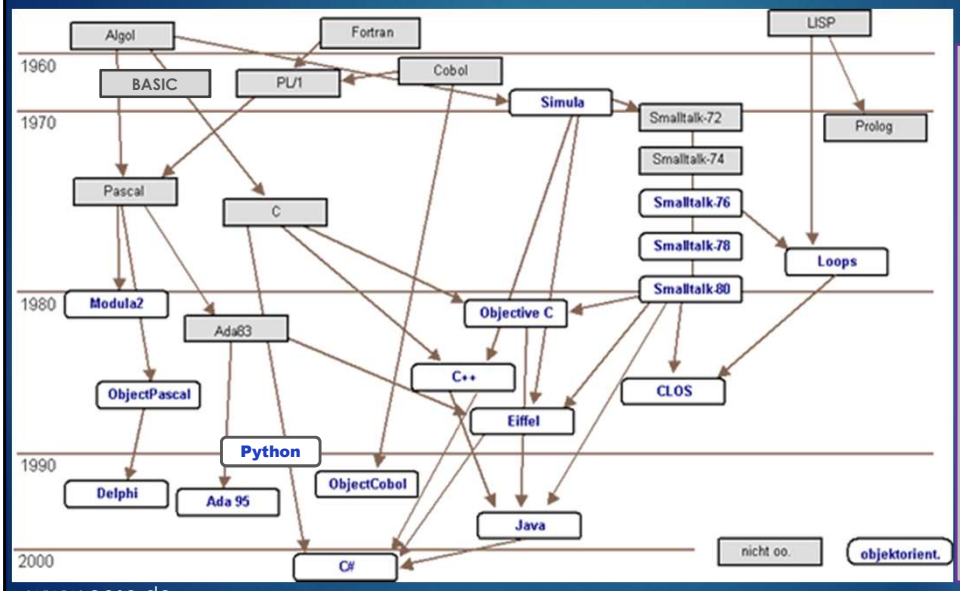
12

- 1940-es évek – első programok,
- 1968 – szoftver krízis,
- 1990 – HTML 1.0
- 1991 – Linux
- 1996 – SUN Java 1.0,
- 2002 – C#1.0 .NET,
- 2007 – iOS 1
- 2008 – Android
- 2012 – Windows 8
- 2015 – Windows 10

A szoftver alkotó elemek (nyelvek, rendszerek, technológiák egyre gyorsabban jelennek meg.

A programozási nyelvek megjelenése

13



www.oose.de

Kemény János György
(1926-1992)
Matematikus
Budapesten született, 1940-től Amerikában élt és alkotott. A BASIC programozási nyelv megalkotója, az időosztásos operációsrendszer bevezetője.

Webalkalmazások

14

- Stand alon → hálózat → server és egyedi kliens → HTML, szerver és standard kliens (browser) → szerver és egyedi kliens, API (browser kicsi, a mobil személyes)
- Statikus tartalom - Szöveges weblapok (1990-95) több, linkekkel összekapcsolt tipikusan szöveges fájl (oldal)
- Dinamikus oldalak – Java, XML – számítási képesség – program töltődik le (kártékony is lehet)
- Adatbázisok használata, intenzív média tartalmak (video) – számítási kapacitás (kliensek teljesítménye, grafikai tulajdonságok)
- A szöveges tartalmak visszatérése → fogyatékkal élők részére, felolvasó programok

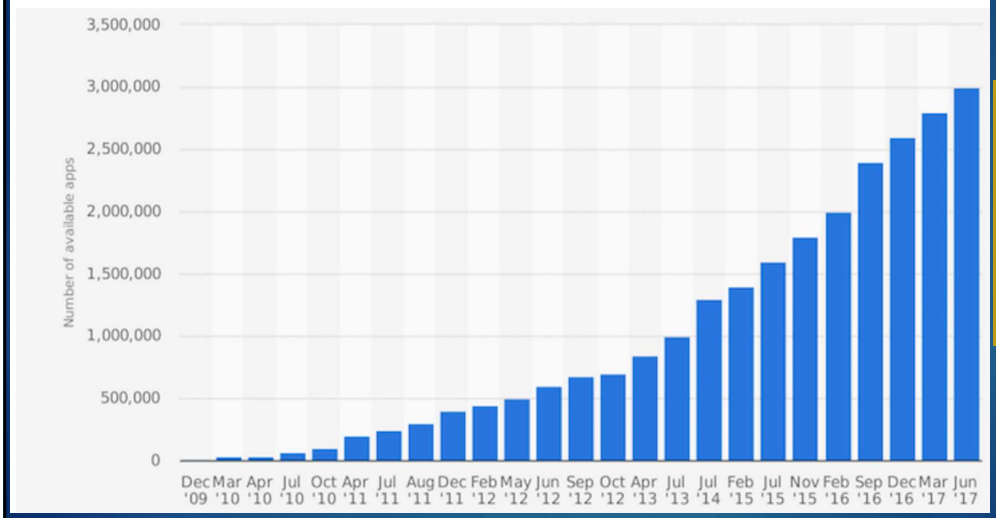
Mobil alkalmazások

15

- A mobil erőforrások rohamos fejlődése
  - Processor (Octa-core (4x2.7 GHz & 4x1.8 GHz))
  - Érzékelők (Iris sensor, Pressure sensor, Accelerometer, Barometer, Fingerprint sensor, Gyro sensor, Geomagnetic sensor, Hall sensor, HR sensor, Proximity sensor, RGB Light sensor)
  - Audió-video nagyon jó minőségben, teljes konnektivitás
- Egyedi applikációk, authentikáció megoldva,
- A lehetőségek sokkal nagyobbak, mint a PC esetében
- App Store-okból nagyon sok minden letölthető
- SW library, resources

16

Number of available applications in the Google Play Store, 2009-2017



Az alkalmazások megjelenésének sebessége az elmúlt 8 évben közel exponenciális.

http://www.businessofapps.com/data/app-statistics/#2 – 2019.02.11.

Felhő-alapú alkalmazások

17

- A komplexitás és az erőforrás-igény rohamos növekedése → felhő szolgáltatások.
- Infrastruktúra-szolgáltatás (IaaS) - virtuális gépek, tárhely, hálózat, operációs rendszer igénybevétele használatalapú díjfizetéssel.
- Platformszolgáltatás (PaaS) - igény szerinti környezet bérlése alkalmazások fejlesztéséhez, teszteléséhez, terjesztéséhez és felügyeletéhez.
- Szoftverszolgáltatás (SaaS) – alkalmazásaink tehetők elérhetővé az interneten keresztül, a szolgáltató üzemelteti és kezeli az alkalmazásokat, a háttér-infrastruktúrát, gondoskodik a szoftverfrissítések, a biztonsági javítások és egyéb karbantartási feladatok elvégzéséről is.

Felhő-alapú alkalmazások

18

- Sokkal gyorsabb rendelkezésre állás (nem kell beszerezni) dinamikusan bővíthető, használat-alapú fizetés, garantált rendelkezésre állás.
- Technikai és üzleti kiszolgáltatottság.
- Rendelkezésre állási biztonság, adatvédelem (lopás, adat-szivárgás), adatvesztés kérdései.
- Rohamosan terjedő technológiai elem.

A felhőbe könnyen fel lehet menni, de kijönni nem lehet soha!

2., A szoftver fejlesztési folyamat

19

- Fejlesztési folyamat → tervezett aktivitások összessége

- Afejlesztési folyamat elemei
  - Analízis (Communication)
  - Tervezés (Planing)
  - Modellezés (Modelling)
  - Implementáció (Construction)
  - Átadás (Deployment)

20

- Kísérő tevékenységek
  - Szoftver projekt irányítás és követés
  - Kockázat menedzsment
  - Szoftver minőségbiztosítás
  - Technikai áttekintés
  - Projekt és szoftver mérés
  - Szoftver konfiguráció menedzsment
  - Újrafelhasználás menedzsment
  - Termék menedzsment

2.1., Szoftver folyamat modellek

21

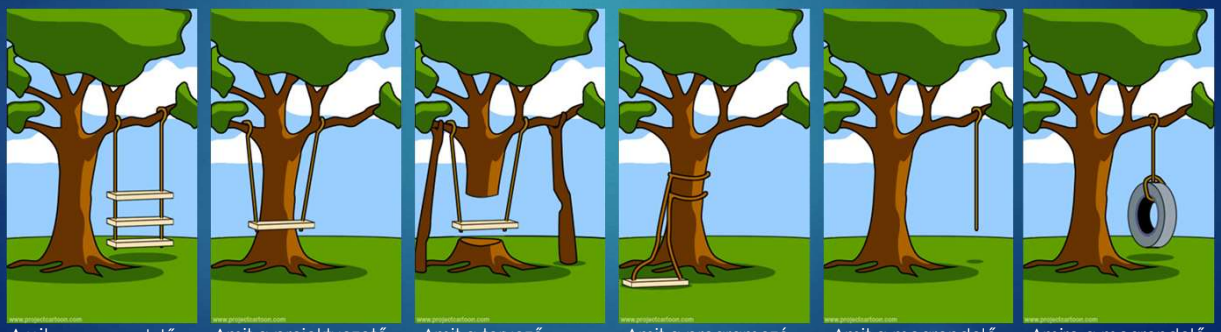
- Afejlesztési folyamat modell típusai
  - 2.2. Prescriptive folyamat modellek
    - 2.2.1. Vízesés modell
    - 2.2.2. V modell
  - 2.3. Inkrementális folyamat modellek
    - 2.3.1. RAD modell
  - 2.4. Evolúciós folyamat modellek
    - 2.4.1. Prototípus modell
    - 2.4.2. Spirál modell
  - 2.5. Konkurens folyamat modellek

22

- Afejlesztési folyamat modell típusai (folytatás)
  - 2.6. Speciális folyamat modellek
    - 2.6.1. Komponens-alapú fejlesztés
    - 2.6.2. Formális módszerek modell
    - 2.6.3. Aspektus-orientált szoftverfejlesztés
    - 2.6.4. Egységes (Unified) folyamat
  - 2.7. Agilis módszerek
    - 2.7.1. XP
    - 2.7.2. SCRUM
  - 2.8. Agilis vs. Tradicionális módszertanok

2.2. Prescriptive folyamat modellek

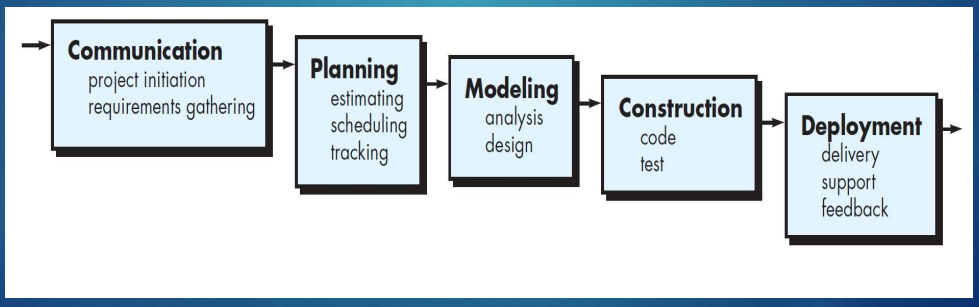
- Szigorúan meghatározott, szekvenciális modellek.
- Visszalépési lehetőséget korlátozottan tartalmaz.
- Nehezen kezeli a projektben a változásokat.
- Érzékeny a kezdeti specifikáció minőségére.



Amit a megrendelő elmondott | Amit a projektvezető megértett belőle | Amit a tervező megtervezett | Amit a programozó leprogramozott | Amit a megrendelő-nek átadtak | Amire a megrendelő-nek szüksége lett volna

2.2.1. Vízesés modell

- Hagyományos szemléletű, erősen szeparált fázisokat tartalmazó, szekvenciális modell.
- Az eredeti vízesés modellt 1970-ben W. W. Royce publikálta „Managing the Development of Large Software Systems" címmel.



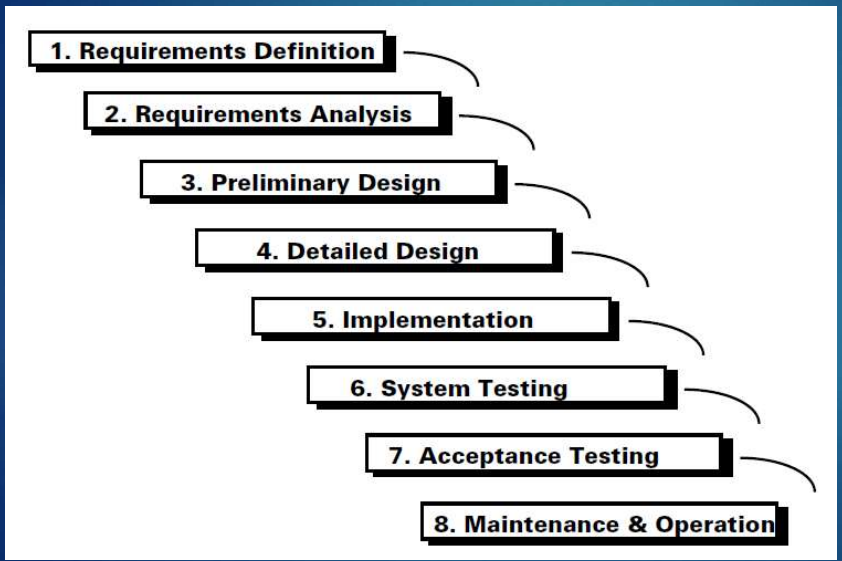
R. S. Pressmann, B. R. Maxim: Software engineering : a practitioner's approach Eighth Edition

25

- Jellemzői:
  - szisztematikus, szekvenciális, top-down modell
  - a hagyományos mérnöki szemléletet követi
  - leginkább elterjedt (legrégibb) modell
- Előnyei:
  - modellt ad a különböző fázisokhoz tartozó módszerek elhelyezésére
  - logikus, könnyen érthető, kézenfekvő modell
  - sok tapasztalat halmozódott fel
- Problémái:
  - a valós projektek ritkán követnek szekvenciális modellt
  - nehezen valósítható meg az iteráció
  - az egész modell a specifikáció minőségétől függ
  - a projekt elején meglévő kezdeti bizonytalanságot nem tudja kezelni
  - nagyon későn lát a megrendelő működő programot

26

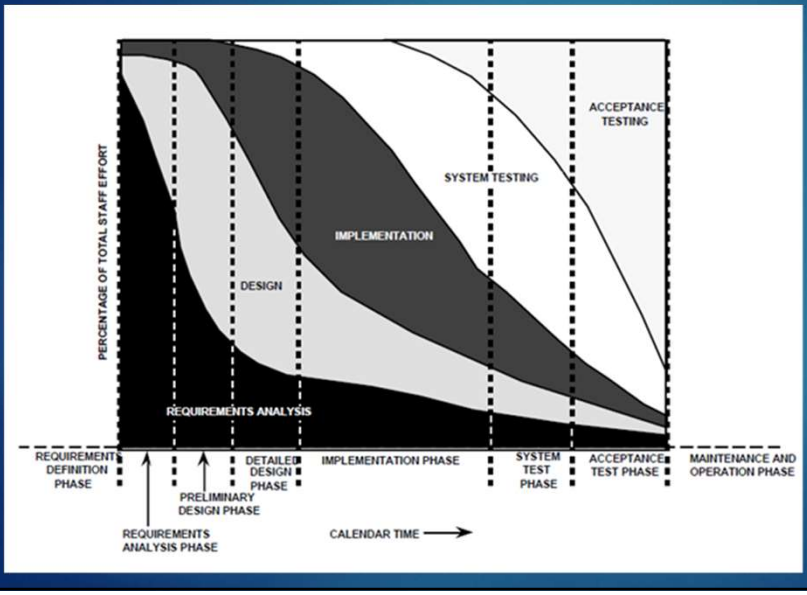
NASA Goddard Space Flight Center
Software Engineering Labor ajánlása



http://sel.gsfc.nasa.gov

NASA Goddard Space Flight Center
Software Engineering Labor ajánlása

27



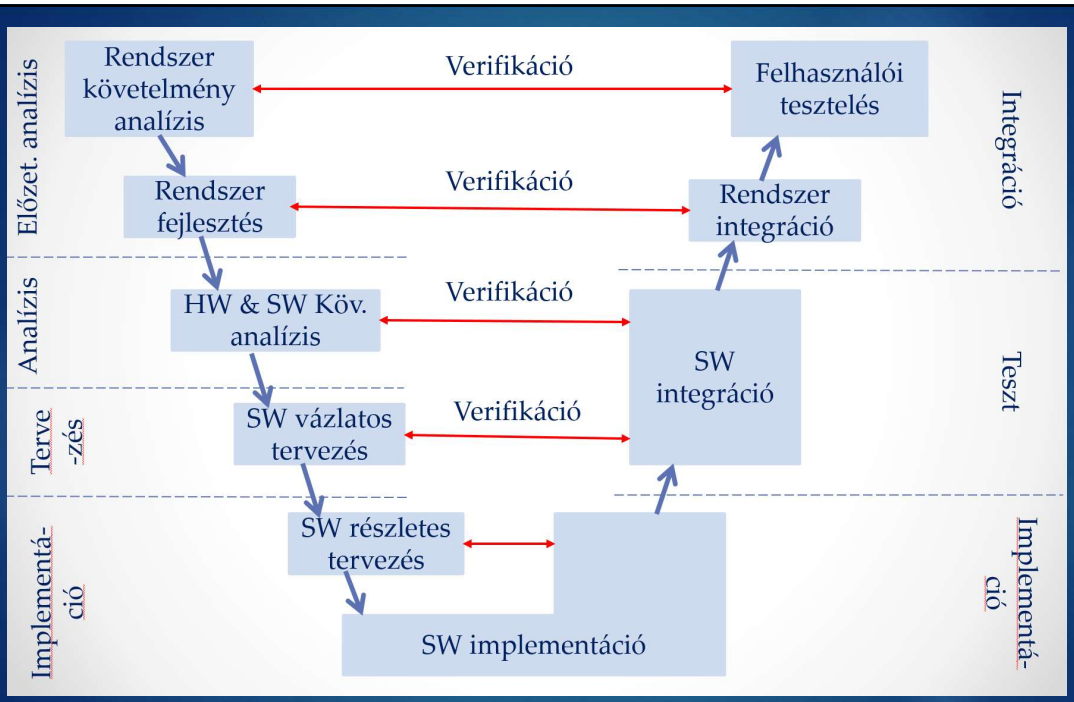
http://sel.gsfc.nasa.gov

2.2.2. „V" modell

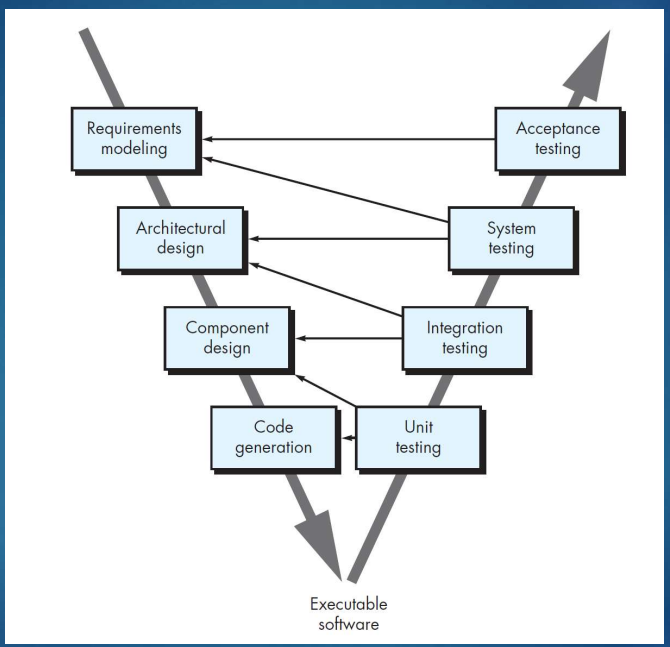
28

- A német védelmi minisztérium által kifejlesztetett modell, melyet 1991-ben kötelezően alkalmazandó módszerként vezettek be a német hadsereg szoftverfejlesztéseinél.
- A vízesés modell továbbfejlesztéseként fogható fel a tesztelés hangsúlyossá tételével.
- Német vállalatok előszeretettel alkalmazzák. 1999-től Ausztriában is bevezették a kormányzati szektorban
- Szigorú dokumentálás mellett részletesen és pontosan tartalmazza a teendőket minden egyes fázisra.

29



30



R. S. Pressmann, B. R. Maxim: Software engineering : a practitioner's approach Eighth Edition

2.3. Inkrementális folyamat modell

31

- Az inkrementális fejlesztés esetén a rendszert kisebb egységekre (inkremensekre) bontjuk, és ezáltal a specifikáció, a tervezés, az implementálás kis inkrementális lépésekben valósul meg.
- Az inkremensek a felhasználóval egyeztetett, előre tervezett, jól meghatározott részek.
- Ha egy inkremens elkészül, átadásra kerül, azt a megrendelő akár használatba is veheti, teszteli.
- A módszert Harlan D. Mills ajánlotta 1980-ban
- Cél a komplexitás csökkentése, a nagy rendszer felbontása, a változások, félreértések miatt szükségessé váló átdolgozások számának csökkentése a fejlesztési folyamatban.
- A módszer lehetőséget ad a megrendelőnek bizonyos - a követelményekkel kapcsolatos - döntések későbbre halasztására.

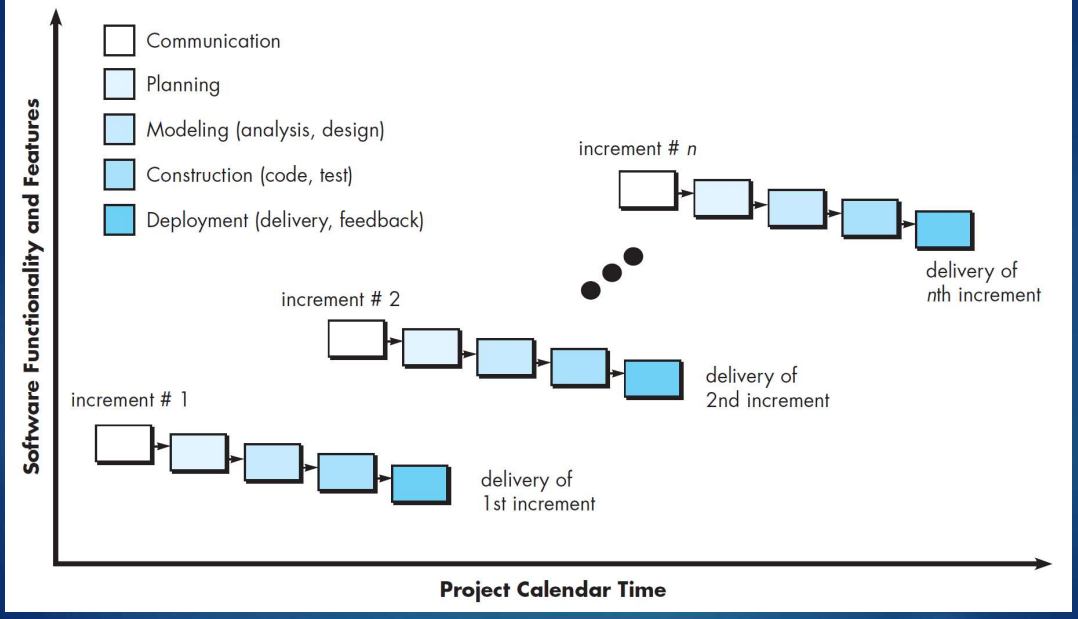
Prof. Dr. Harlan D. Mills
State University of Florida, Iowa, Princeton

32

Inkrementális

Iteratív

33



R. S. Pressmann, B. R. Maxim: Software engineering : a practitioner's approach Eighth Edition

34

- Az inkrementális fejlesztés előnyei
  - A szoftver már menetközben használhatóvá válik a megrendelő számára (kritikus követelmények teljesülnek először)
  - A megrendelők használhatják a korábbi inkremenseket, mint prototípusokat a későbbi inkremensekhez
  - Kisebb a kockázata a projekt kudarcának, biztonságos fejlesztés
  - A legkritikusabb funkciókat teszteljük legtöbbet (azok készülnek el először)

2.3.1. RAD modell

35

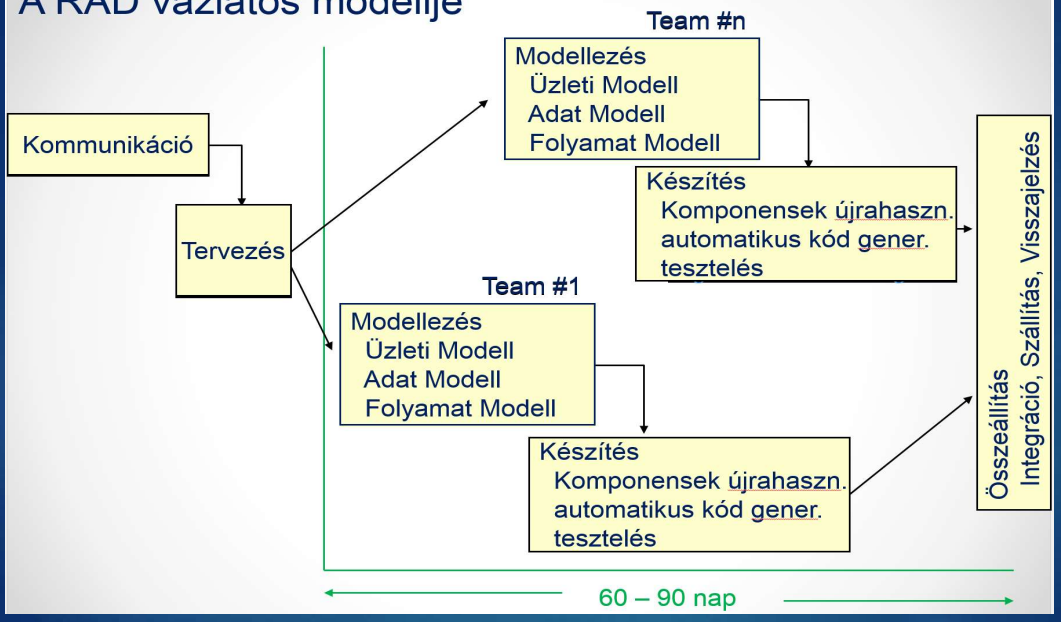
- Rapid Application Development
- J. Martin publikálta 1991-ben
- Inkrementális modell
- Kifejezetten rövid fejlesztési ciklus (60-90 nap)
- „High-speed" adaptációja a vízesés modellnek
- A nagy sebesség a „component based construction" , a „software reuse", az „automatic code generation" technikák alkalmazásával érhető el.

36

- Különböző teamek dolgoznak párhuzamosan a tervezés fázisától a különböző funkciókon
- A team tagjainak sokoldalúnak kell lennie
- A megrendelő szakemberei is részt vesznek a teamek munkájában
- Modern fejlesztői környezetet használ
- RAD-et támogató tool-ként hirdetik magukat:
  - Microsoft Visual Studio .NET
  - Sun Java Studio Creator
  - BEA Web Logic Workshop
  - Borland C# Builder
  - IBM WebSphere Studio Application Developer

A RAD vázlatos modellje

37



R. S. Pressmann: Software engineering : a practitioner's approach Seventh Edition

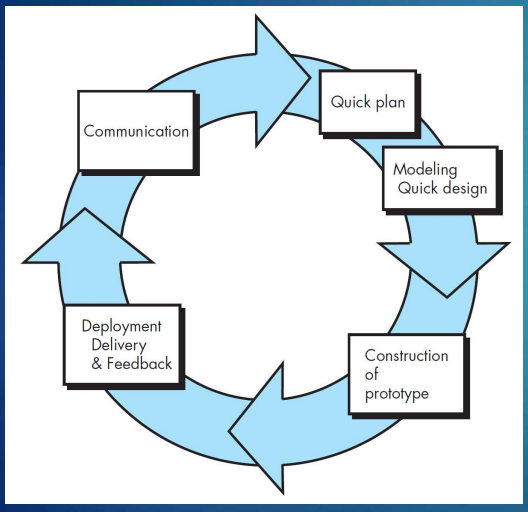
2.4. Evolúciós folyamat modellek

38

- A fejlesztés közben a követelmények változását a szoftver újabb, egyre komplexebb verziói követik
- Iteratív megközelítés érvényesül
- A folyamat nehezen menedzselhető
- Erőforrás és határidőbecslés nehézkes
- A minőségbiztosítás nehezen valósul meg

2.4.1. Prototípus modell

39

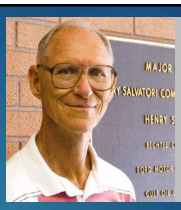


- Gyors fejlesztési folyamat
- A megrendelő intenzív bevonása
- A termék jól illeszkedik a megrendelő elvárásaihoz
- Igazából csak a követelmények meghatározásához használható
- Ha így fejlesztünk terméket, az nem követi a szoftvertechnológia elveit
- Interfész prototípus → később fel-használható

R. S. Pressmann, B. R. Maxim: Software engineering : a practitioner's approach Eighth Edition

2.4.2. Spirál modell

40

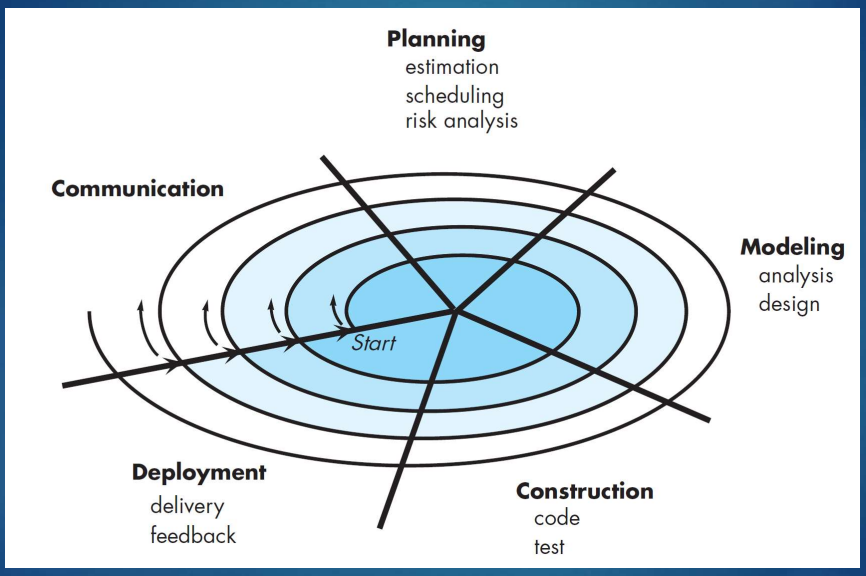


Prof. Dr. Barry Boehm University of Souther California

- Prof. Dr. Barry Boehm publikálta 1988-ban
- A valós fejlesztést jól követő modell, amely kifejezetten jól használható bonyolult, nagy projekteknél
- Az alapgondolata eltér a többi modellétől, itt a szoftverfolyamatot egy spirálként értelmezzük, nem pedig tevékenységek és a közöttük lévő előre-, illetve visszalépések sorozataként.
- A spirál minden egyes körbefordulása a szoftverfolyamat egy-egy fázisát reprezentálja.
- A spirál a tetszőleges számú iterációt is szimbolizálja
- Tartalmazza a kockázat-kezelést is, ami a valós projektek szerves része
- Magába integrálhat más modelleket, (pl. prototípus)

A spirál modell

41



R. S. Pressmann, B. R. Maxim: Software engineering : a practitioner's approach Eighth Edition

2.5. Konkurens folyamat modellek

42

- A fejlesztési folyamat során iteratív és konkurens elemeket egyaránt alkalmaznak
- A párhuzamos modellezés bármely típusú szoftverfejlesztésre alkalmazható (események, tevékenységek)
- A szoftverfejlesztési tevékenységeket, (a műveleteket és a feladatokat) nem eseménysorozatként kezeli, hanem folyamathálózatként
- A hálózat minden tevékenysége vagy feladata létezik egyidejűleg más tevékenységekkel vagy feladatokkal együtt. A folyamathálózat egyik pontján keletkezett események az egyes tevékenységekhez kapcsolódó állapotok közötti váltásokat eredményezik.

2.6. Speciális folyamat modellek

43

- Az előzőekben megismert hagyományos modellekkel szemben a speciális modellek mindegyike egyedi aspektusokat tartalmaz.
- Egy részük a komplexitás kezelésének speciális módját adják.
- Egy másik részük speciális követelmények (pl. bizonyíthatóan helyes működés) teljesítését helyezi előtérbe.

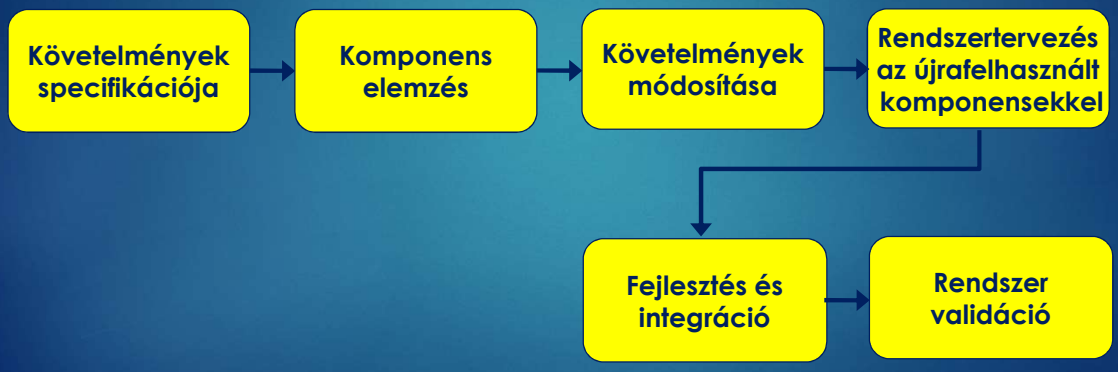
2.6.1. Komponens-alapú fejlesztés

44

- A komponens alapú fejlesztés esetén az elkészítendő szoftvert újrafelhasználható komponensekből igyekszünk felépíteni.
- A fejlesztési idő nagy mértékben lerövidíthető
- Lényegesen olcsóbbá válik a fejlesztés
- A kifejlesztett rendszer minősége, megbízhatósága jobb, mert leellenőrzött, már sokszor használt elemeket építünk be.
- A felhasználható komponensek függvényében módosulhatnak a követelmények

A komponens alapú fejlesztés modellje

45



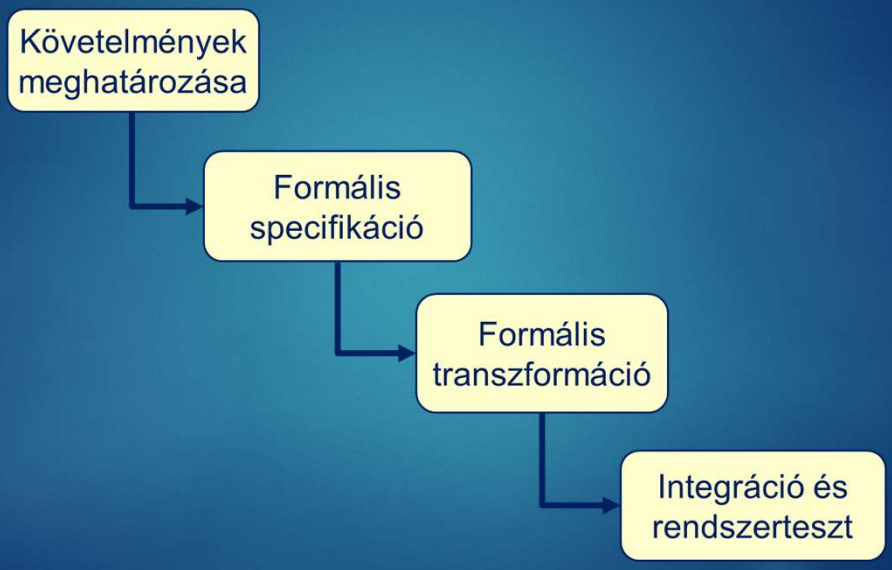
2.6.2. Formális módszerek modell

46

- A formális rendszerfejlesztés jellegében a vízesés modellhez hasonlít (szekvenciális).
- Lényege a specifikáció futtatható formába történő transzformálása formális matematikai eszközökkel.
- A transzformációk korrektek, így könnyű a verifikációjuk, (a program a specifikáció szisztematikus transzformációja).
- Használata nagyon előnyös olyan rendszerek, vagy részrendszerek esetében, ahol elsődleges a rendszer biztonsága, megbízhatósága és annak autentikus bizonyítása.

A formális fejlesztés fázisai

47



2.6.3. Aspektus-orientált szoftverfejlesztés

48

- A fejlesztés súlypontja egyre inkább tolódik el a nagy szoftverrendszerek felé.
- A nagy szoftverrendszerek komplexek, nagyon sok követelményt kell teljesíteniük működésük során.
- Ezen rendszerek a komplexitás kezelhetősége miatt komponensekből épülnek fel.
- A követelmények és a komponensek között nem lehet 1:1 arányú megfeleltetés. (borul a tiszta felelősség elv!)
- Következmény: a követelmények változása esetén több komponens módosítása szükséges, ami kihathat más követelményekre is.

Aspektus-orientált szoftverfejlesztés (AOSD)

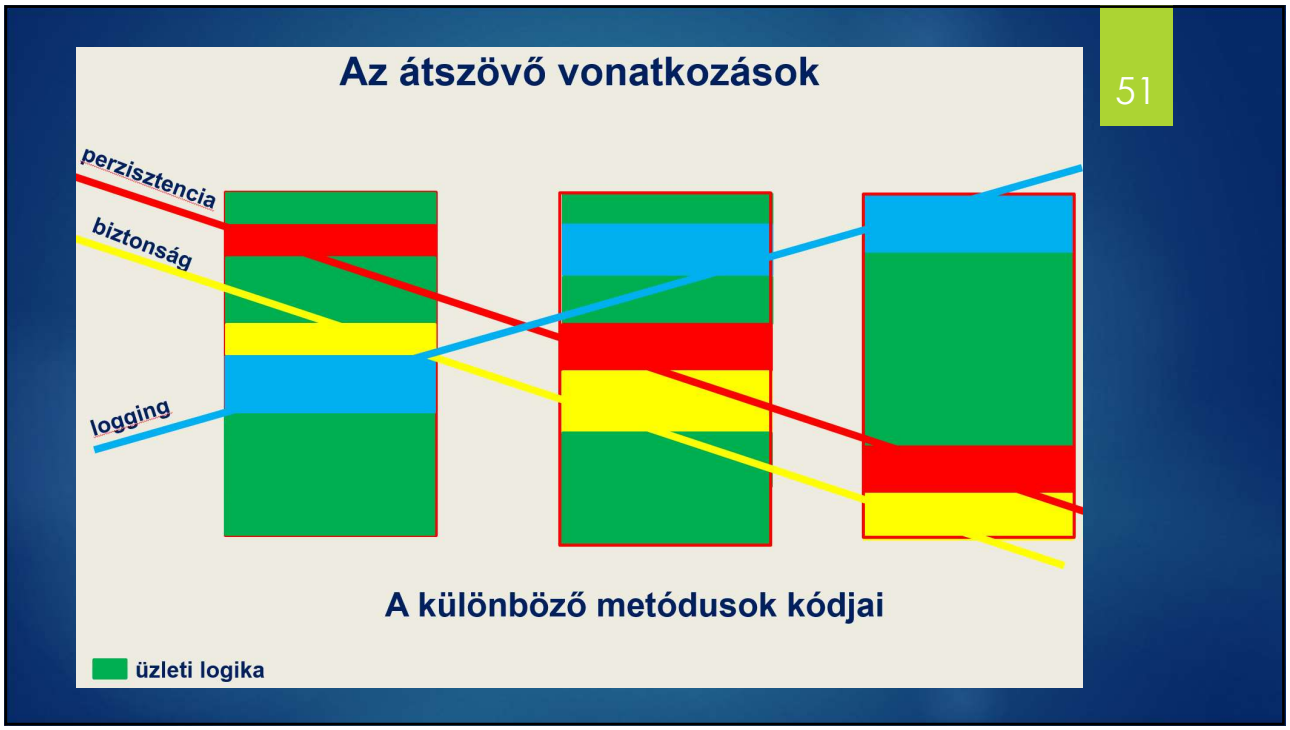
49

- Az eddig alkalmazott absztrakciókhoz (módszerek és objektumok) egy új absztrakciót vezet be: az aspektust.
- **Aspektus:** azon funkciók bezárására szolgál, amelyek keresztezik a rendszer más funkcióit, vagy azokkal együtt létezik (a szerkezet tisztítása, „átnyúló részek kiemelése).
- Az OO-nál magasabb szintű koncepció:
  - OO: módszerek és objektumok
  - AO: módszerek, objektumok és aspektusok

A vonatkozás értelmezése

50

- Amikor egy komplex OO szoftver esetében meghatározzuk az objektumokat, akkor az önálló entitások létrehozására törekszünk (metódusok, adatok, zártság).
- Így azonban „szétszóródik" néhány generális szempont (pl. perzisztencia, hibakövetés, naplózás), ami összetartozik.
- Ezeket a szétszórt, de logikailag összetartozó, a program különböző elemein áthúzódó kódokat nevezzük átszövő **vonatkozásoknak** (crosscutting concerns).



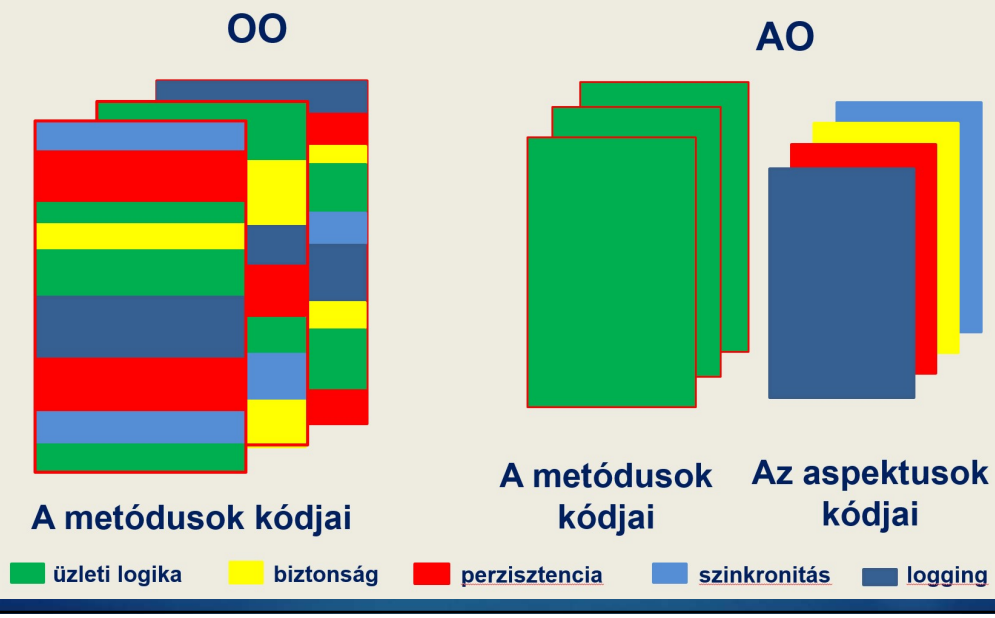
Az Aspektus-Orientált szoftver jellemzői

52

- A vonatkozásokat az aspektusokban valósítjuk meg.
- Lényege, hogy a végrehajtható kód a módszerek, objektumok és aspektusok automatikus összeszövéséből jön létre.
- Az összeszövés módja a program forráskódjában van definiálva.
- Előnye, támogatja a vonatkozások szétválasztását és ezáltal egy tisztább kép alakul ki a rendszerről (módosítás, újrafelhasználhatóság, stb).

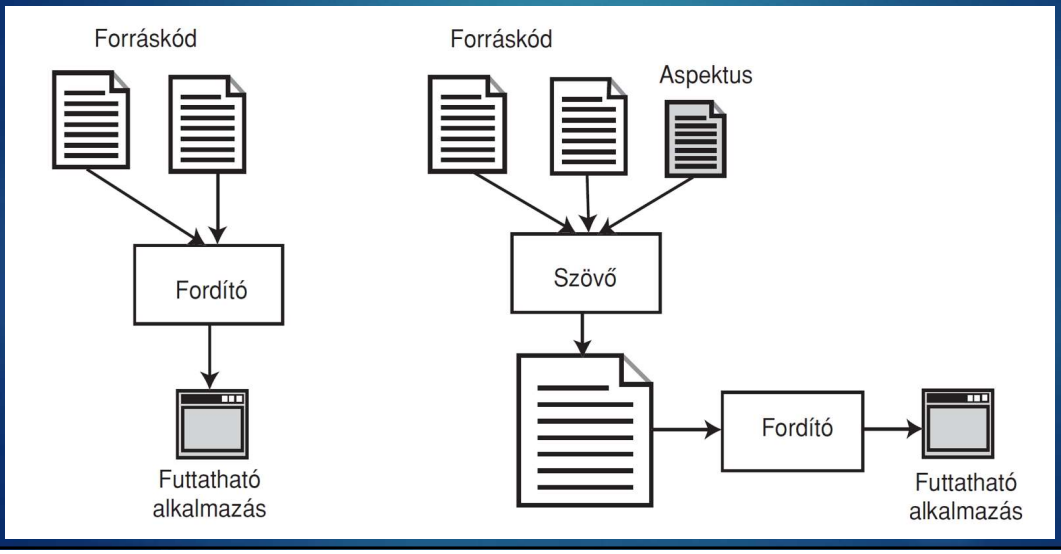
OO és AO rendszerek összehasonlítása

53



Az OO és az AO futtatható alkalmazás elkészítése – az aspektusszövő

54



2.6.4. Egységes (Unified) folyamat

55

- Más néven Rational Unified Process (RUP)
- A Rational Software Corporation támogatásával fejlesztette ki Grady Booch, Ivar Jacobson, és Jim Rumbaugh (The three amigos)
- Az UML-hez jól illeszkedő szoftver-fejlesztési módszertan
- Korábbi módszertanoktól átvesz bevált megoldásokat
- A legjobb gyakorlatokat igyekszik egységes rendszerbe foglalni (best practice)
- Sok eszközzel támogatott (sok cég állt mögé)

A „3 Amigo”
Unified Modeling Language,
Unified Process

Grady Booch
(1955-)
Booch Method
(OOAD), OO Design
Patterns

Ivar Jacobson
(1939-)
Use Case, OOSE,

James Rumbaugh
(1947-)
Object Modeling
Technique

56

A RUP jellemzői

- Iteratív és inkrementális
- Architektúra-centrikus
- Használati eset vezérelt (a középpontban a megrendelővel egyeztetett „Use-Case”-ek állnak)
- Komponensalapú (a rendszer egymással kommunikáló elemekből, komponensekből épül fel)
- Modell szemléletű

A RUP felépítése

- A fejlesztési folyamatot tevékenységekre és fázisokra bontja
- A tevékenységek között megkülönböztet:
  - Alap tevékenységeket
  - Támogató tevékenységeket

Fázisok:

Előkészítés (Inception)

Kidolgozás (Elaboration)

Megvalósítás (Construction)

Átadás (Transition)

57

Alap tevékenységek:

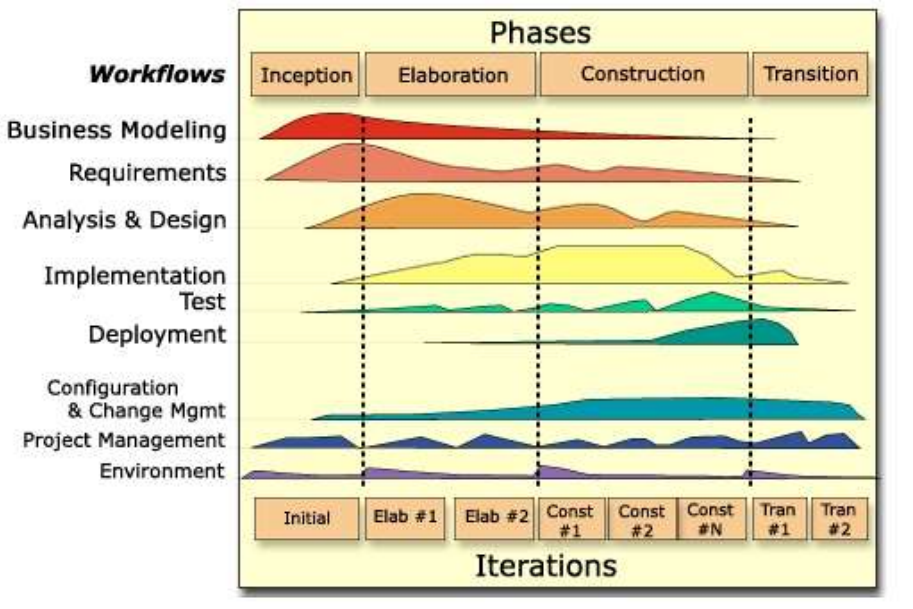
- Üzleti folyamatok modellezése (Business Modeling)
- Követelményelemzés (Requirements)
- Analízis és tervezés (Analysis & Design)
- Implementáció (Implementation)
- Tesztelés (Test)
- Átadás (Deployment)

Támogató tevékenységek:

- Konfiguráció menedzsment és változáskövetés (Configuration & Change Management)
- Projektvezetés (Project Management)
- Fejlesztői környezet biztosítása (Environment)

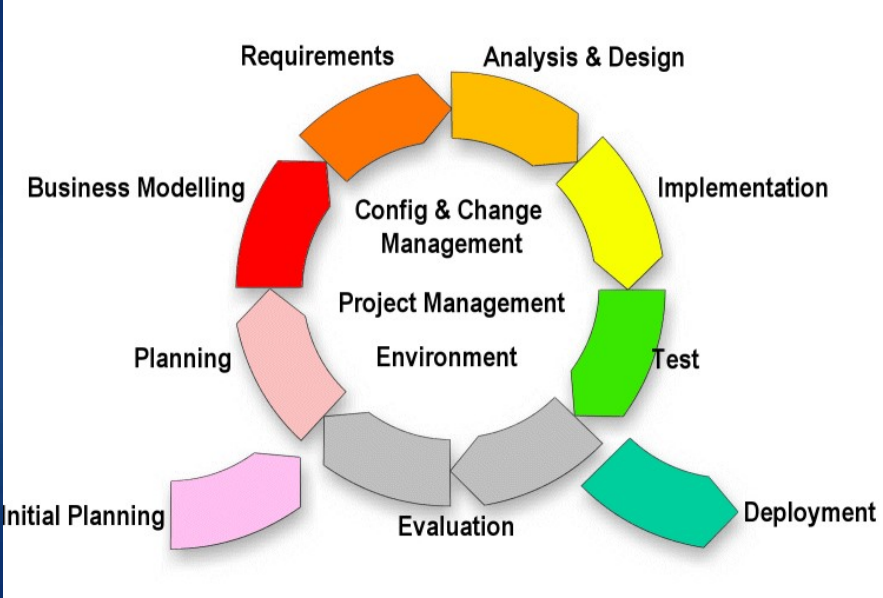
Az UP működése

58



Iteráció a UP-ben

59



2.7. Agilis módszerek

60

2001-ben 17 neves szoftver fejlesztő alapította meg az „Agile Allience”-t és fogalmazta meg a „Manifesto for Agile Software Development”-et.
(www.agilemanifesto.org)

Kiáltvány az agilis szoftverfejlesztésért

61

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 szolgai követésével szemben

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

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Agilis elvek

62

- Első rendű szempont a megrendelő maradéktalan kielégítése
- Flexibilitás a követelmények változásával szemben
- Működő szoftver gyors átadása (inkremensek)
- Az üzleti szakembereknek és a fejlesztőknek napi kapcsolatban kell lenniük
- Motivált szakembereket gyűjts a projekt köré, teremtsd meg a feltételeket a munkájukhoz
- Szorgalmazd a szemtől-szembeni párbeszédet a teamen belül az információ cserére

63

- A haladás legjobb mértéke a működő szoftver
- „Agile process" fenntartható fejlődést ösztönöz
- A fejlesztőknek és a megrendelőnek egy állandó ütemet kell fenntartani a fejlesztési folyamatban
- A kiváló műszaki színvonalra és a jó tervre folyamatosan ügyelni kell
- Egyszerűség – csak az igazán fontos feladat elvégzése
- Az önszerveződő, aktív teamektől származnak a legjobb megoldások (szerkezetekre, tervekre, követelményekre)
- Rendszeres időközönként a teamnek önvizsgálatot kell tartani és viselkedését esetleg módosítani

Agilis elveket követő módszerek:

64

- Extreme Programming (XP)
- Adaptive Software Development (ASD)
- Dynamic Software Development Method (DSDM)
- Feature Driven Development (FDD)
- Scrum
- Kanban
- Crystal
- Agile Modeling (AM)

2.7.1. Extrém Programozás - XP

65

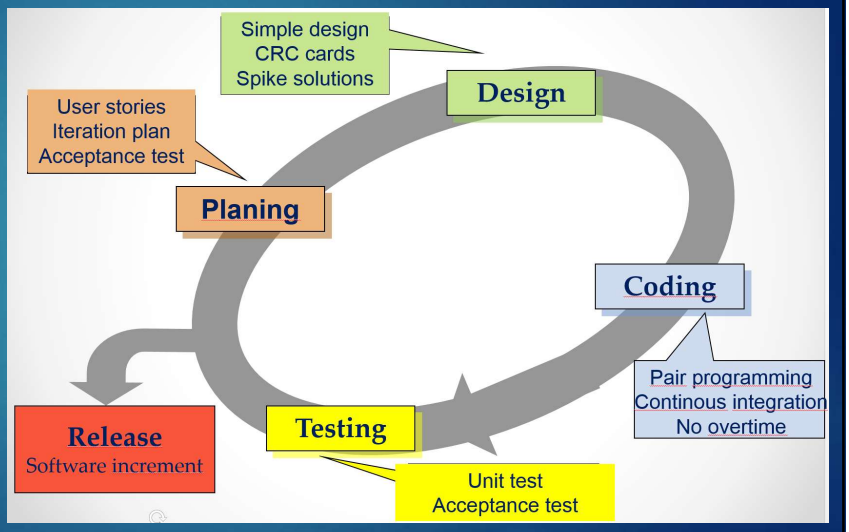
- Kent Beck publikálta először 1999-ben
- Inkrementális megközelítés OO alapon
- Ajánlott szabályok és praktikumok
- 4 alapvető aktivitásból áll:
  - Planning
  - Design
  - Coding
  - Testing

Kent Beck
(1961-)
Szoftverfejlesztő
Az agilis koalíció egyik alapítója, az extrém programozás (XP) és a tesztvezérelt fejlesztés megalkotója (TDD). A Smalltalk és a CRC kártyák népszerűsítője.

Az XP alapvető értékei

66

- Kommunikáció
- Egyszerűség
- Visszacsatolás
- Bátorság
- Tisztelet



Az XP vázlatos modellje

Az XP előnyei

67

- Jól kezeli a követelmények változását
- Költséghatékonyabb a hagyományos módszereknél
- A legjobb megoldásokat ötvözi egy módszertanná
- Nagyobb a megrendelői elégedettség, a megrendelő intenzív bevonása miatt
- Alacsonyabb a kockázati tényezője, mint a hagyományos módszereknek

Az XP hátrányai

68

- A termék minősége erősen függ a megrendelő közreműködésétől (a megrendelők egy része „nem kedveli" az intenzív bevonását a projektbe.)
- Nem terv centralizált, hanem inkább kód centralizált (nehezebb a reengineering és a reuse)
- Nem fektet elegendő hangsúlyt a dokumentáció készítésére

2.7.2. SCRUM

69

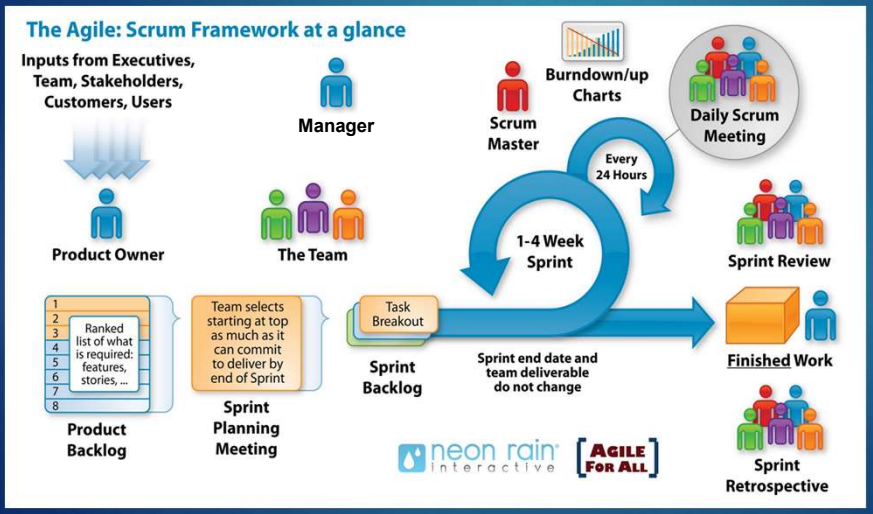
- két japán professzor különböző területeken működő vállalatokat tanulmányoztt, amelyek az átlaghoz képest meglepően gyorsan és jó minőségben fejlesztettek ki új termékeket.
- 1986-ban publikálták a SCRUM-ot
- A legjobb gyakorlatokat ötvözték a SCRUM-ba.

scrum a rugby-ben (dulakodás)

A SCRUM felépítése és működése

70

Szerepkörök:

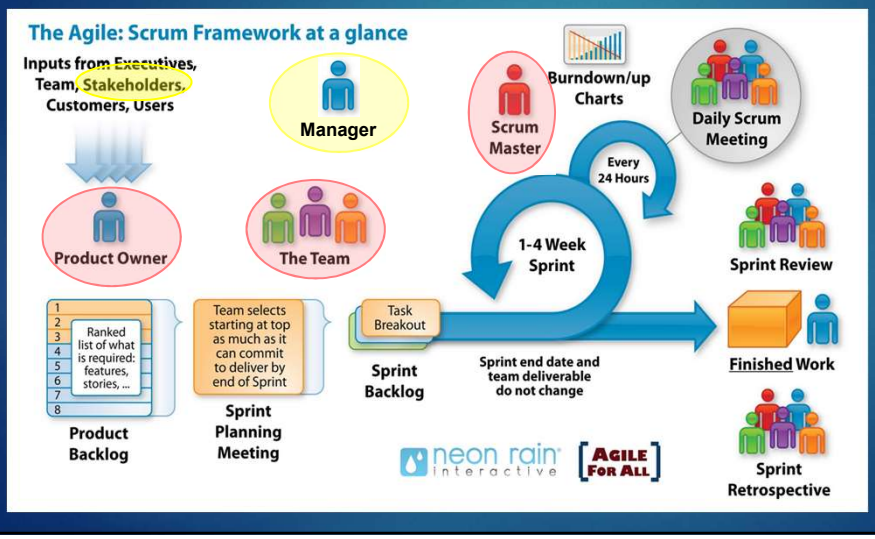


Disznók:
elkötelezettek, szívükön viselik a projektet, mindent bele adnak

Csirkék: a projekt haszonélvezői, érdekeltek a sikerben, de „nem halnak bele" ha mégse sikeres.

A SCRUM felépítése és működése

71



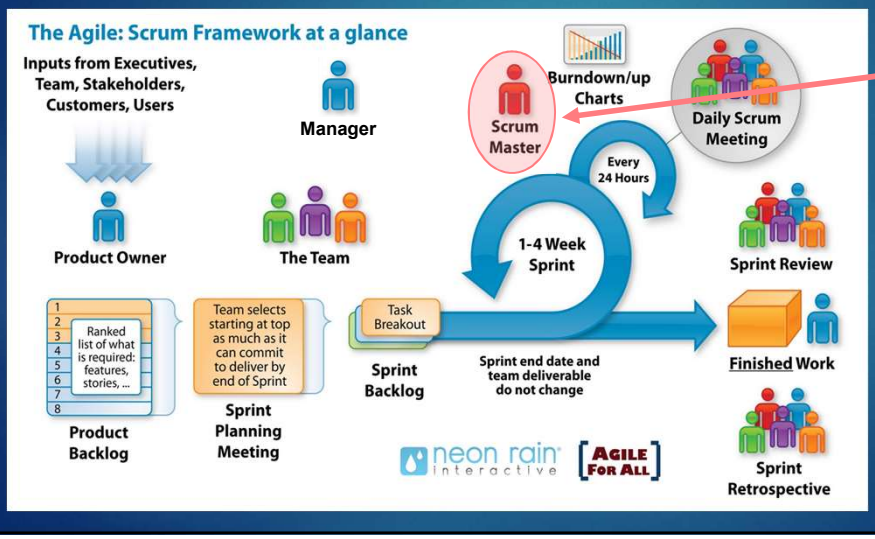
Szerepkörök:

Disznók:
- Scrum Master
- Product Owner
- Team

Csirkék:
- Stakeholders
- Manager

A SCRUM felépítése és működése

72

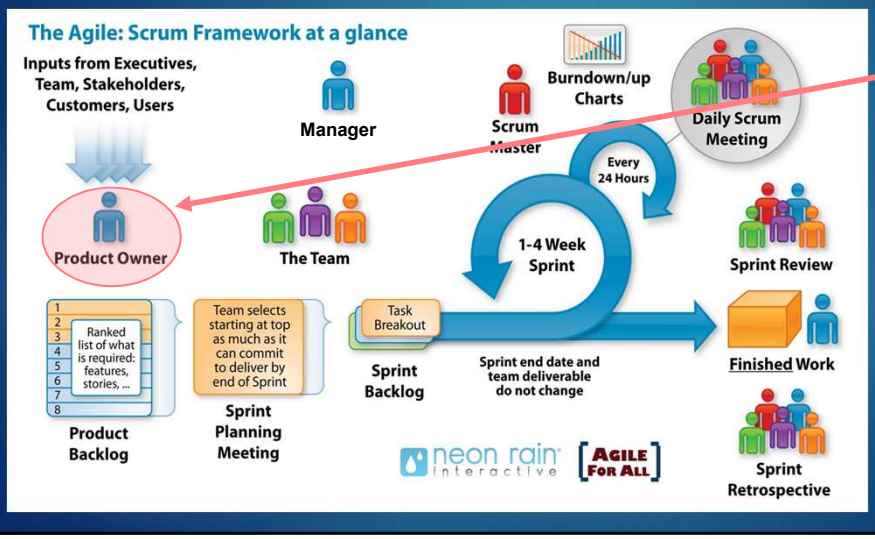


Szerepkörök:

Scrum Master:
- szerepe hasonló a projekt menedzse-réhez,
- felügyeli a folyama-tokat,
- betartatja a SCRUM szabályait
- segít az akadályok elhárításában,
- konfliktust vállal fel

A SCRUM felépítése és működése

73



Szerepkörök:

Product Owner:

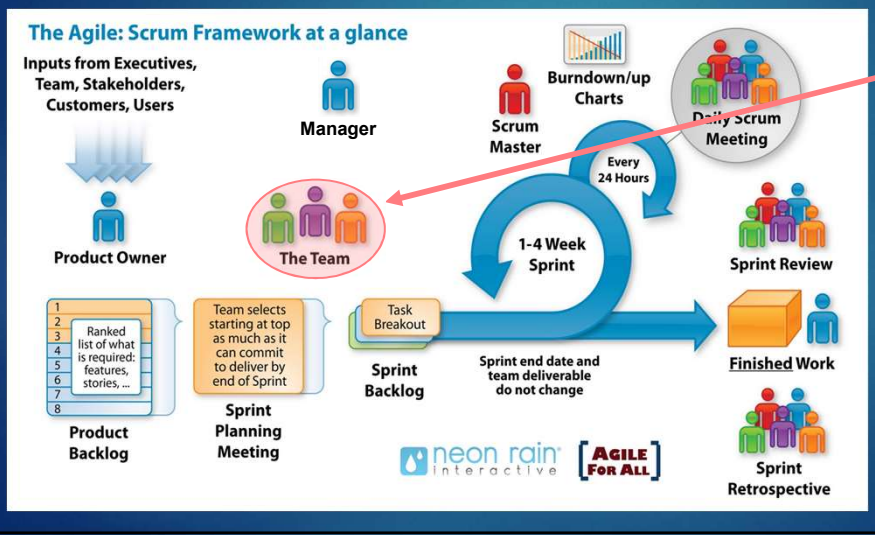
a megrendelő szerepét tölti be, képviseli az érdekeit (lásd XP)

Ügyel arra, hogy a team azt a részt (US) fejlessze, amelyik a legfontosabb a megrendelőnek.

(a User Story-kat priorizálja a PB-ban)

A SCRUM felépítése és működése

74



Szerepkörök:

Team:

végzik a tényleges fejlesztést,

általában 5-9 fő alkot egy teamet,

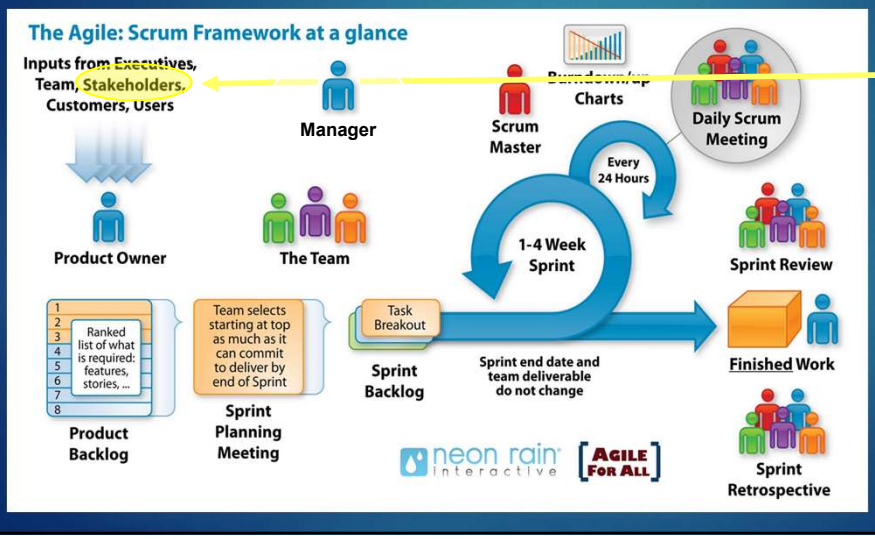
(elemzők, fejlesztők, programozók, tesztelők),

A team felelős a „sprint"-re bevállalt feladatok elvégzéséért,

A SCRUM felépítése és működése

75

Szerepkörök:



Stakeholders:

Az üzleti élet szereplője (megrendelő, vevő, kereskedő)

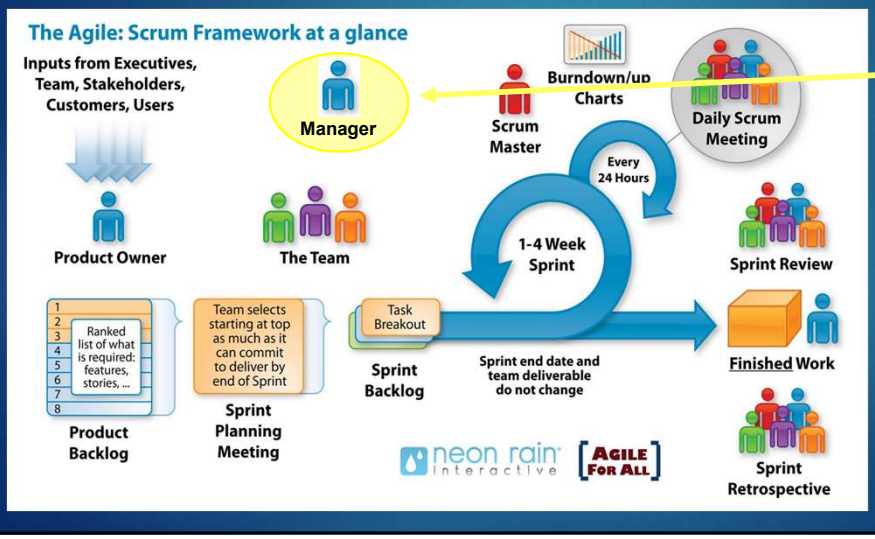
Igénye, elvárásai határozzák meg a terméket (a fejlesztési folyamat-hoz semmi köze)

a Sprint Review során kap szerepet

A SCRUM felépítése és működése

76

Szerepkörök:

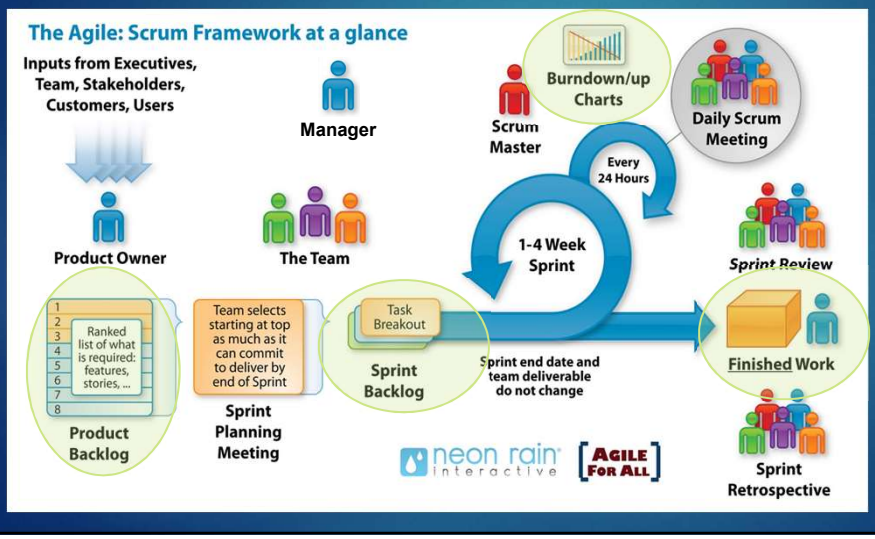


Manager:

általánosságban véve a projekt menedzselését végzi, feladata a SCRUM betartatása, a fejlesztés körülményeinek biztosítása (tárgyi feltételek, szolgál-tatások, pénz)

A SCRUM felépítése és működése

77

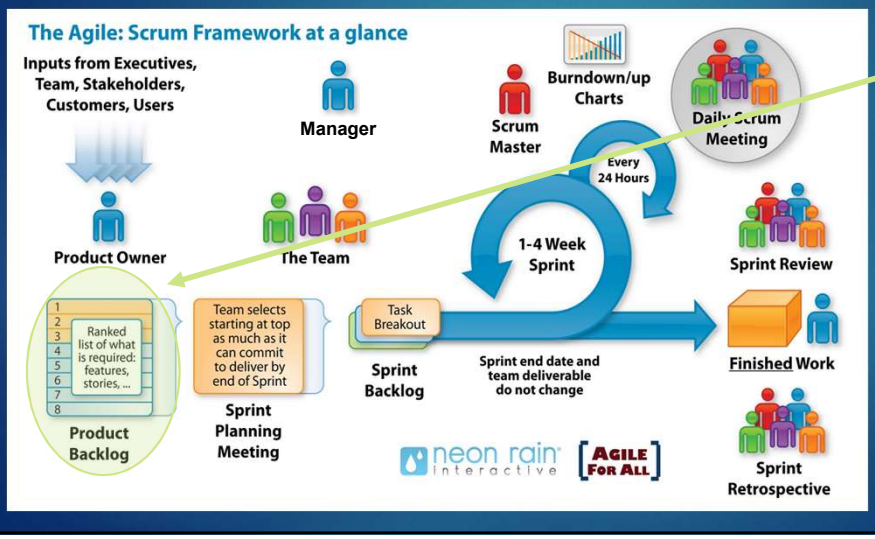


Elemek:

Product Backlog
Sprint Backlog
Burndown/up Chart
Finished Work

A SCRUM felépítése és működése

78



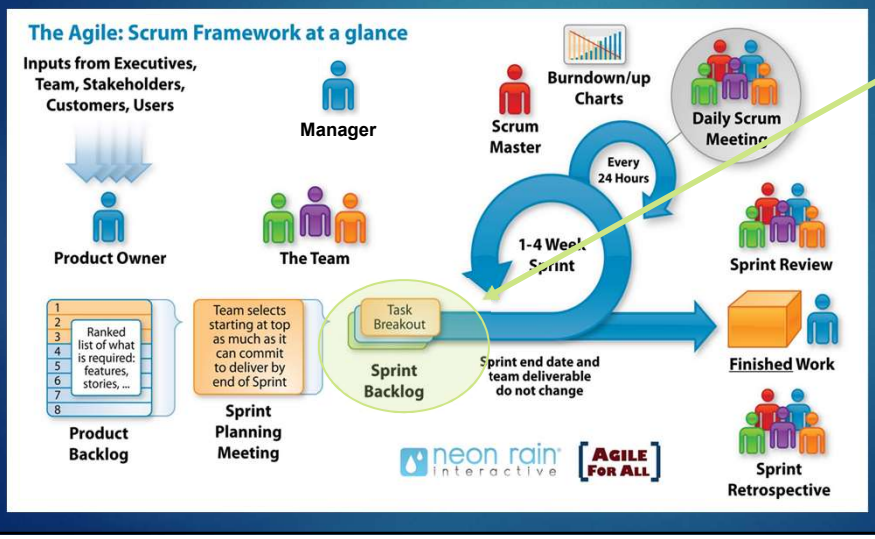
Elemek:

Product Backlog:
a termék-teendők listája, az elkészítendő „sztorikat" tartalmazza,
a Product Owner helyezi el és priorizálja a sztorikat,
az előrébb lévő sztoriknak nagyobb az üzleti értéke.

A SCRUM felépítése és működése

79

Elemek:



Sprint Backlog:

futam-teendő lista,

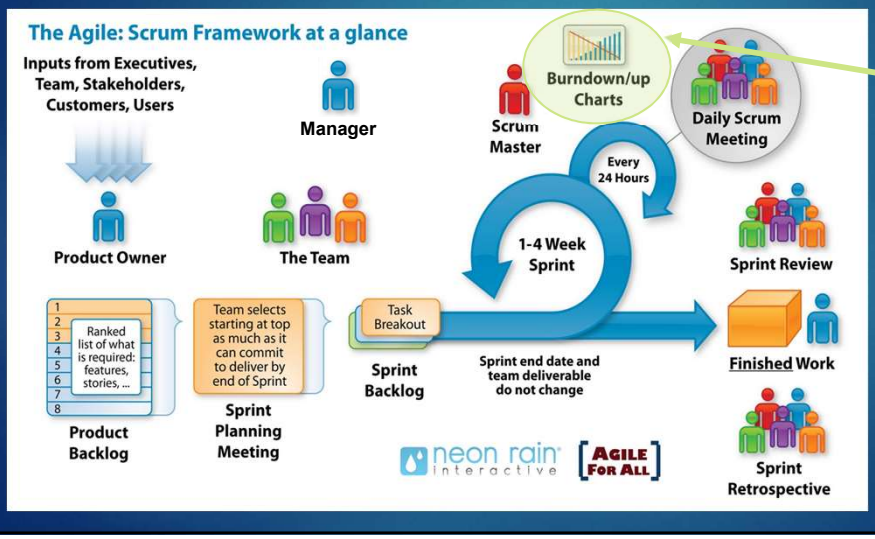
az aktuális sprintre bevállalt sztorikat tartalmazza,

a sztorikat tovább bontják taszkokra, ezeket vállalják el a tagok a Daily Meeting-ek során.

A SCRUM felépítése és működése

80

Elemek:



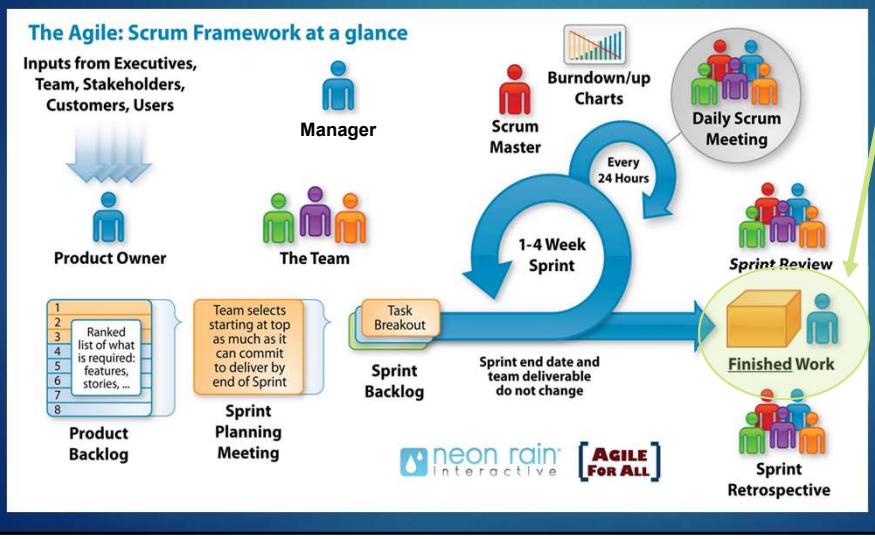
Burndown/up Chart:

Napi Eredmény diagram,

megmutatja, hogy hogyan halad a csapat az aktuális sprinten belül, az eredeti ütemezés-hez képest.

A SCRUM felépítése és működése

81



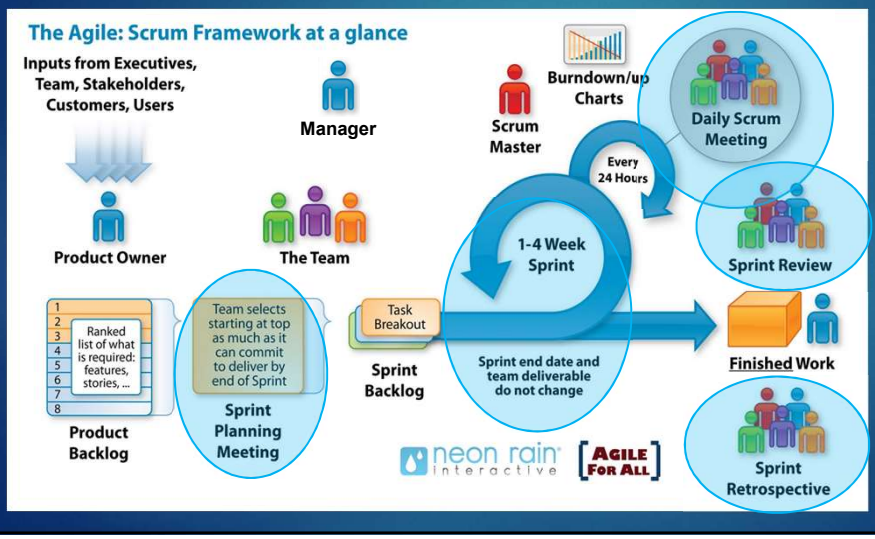
Elemek:

Finished Work:

A sprintek eredményeiből összeállt termék.

A SCRUM felépítése és működése

82



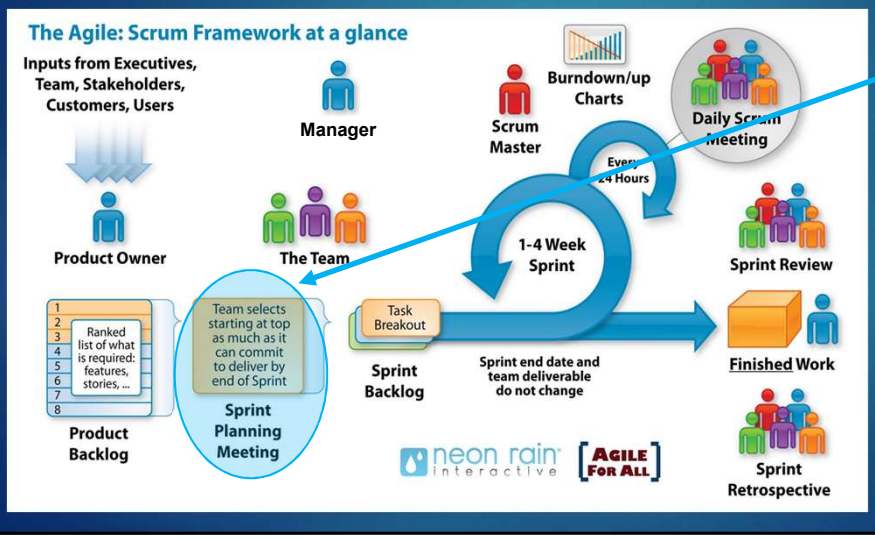
Lépések:

Sprint Planning Meeting
Sprint
Daily Meeting
Sprint Review
Sprint Retrospective

A SCRUM felépítése és működése

83

Lépések:



Sprint Planning Meeting:

futamtervező meg-beszélés,

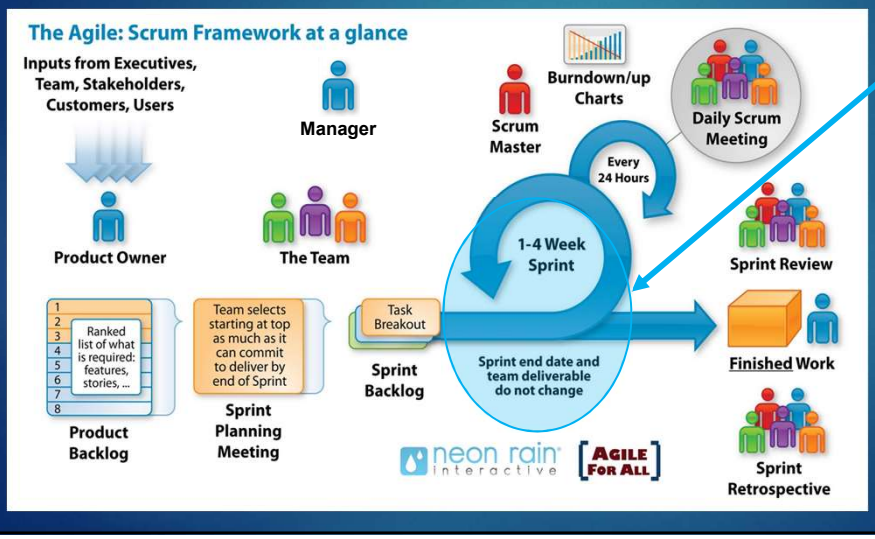
eldől, hogy melyik team melyik sztorit vállalja a sprintre,

a Product Owner-rel megbeszélik a megrendelő elvárásait.

A SCRUM felépítése és működése

84

Lépések:



Sprint:

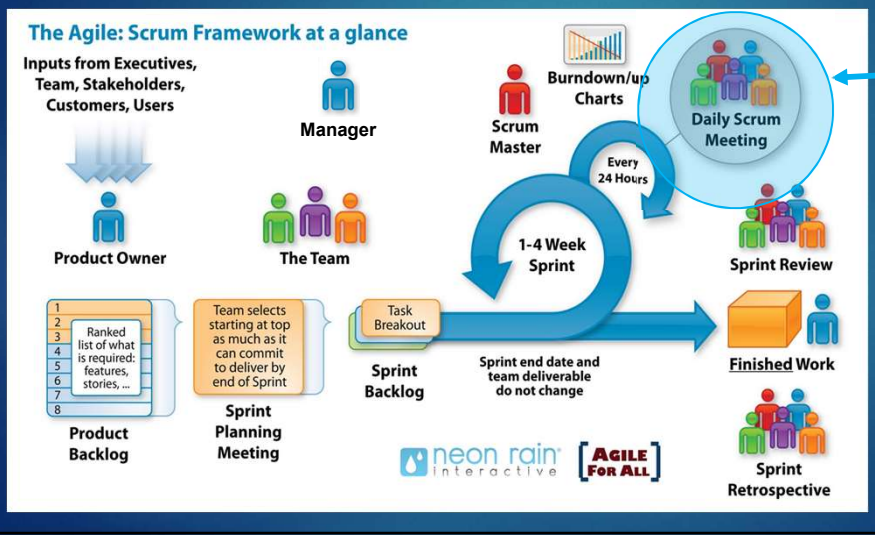
futam, rögzített hosszúságú fejlesztési szakasz, (1-4 hét),

a Scrum iterációs ciklusa, amit addig ismételnek, amíg a Product Backlog-ból el nem tűnnek a megoldásra váró felhasználói sztorik.

A SCRUM felépítése és működése

85

Lépések:



Daily Meeting:

max. 15 perces megbeszélés minden reggel (stand up),

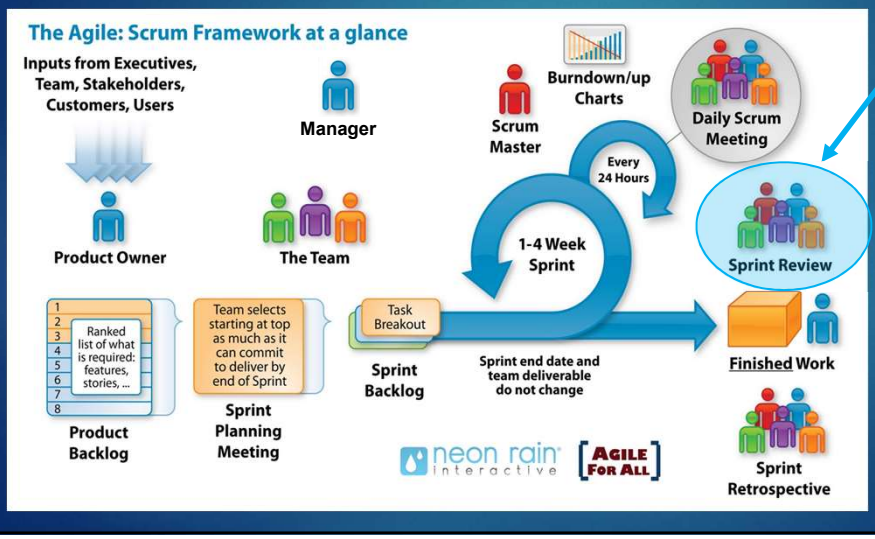
Scrum Master + minden csapattag vesz részt,

3 kérdésre kell minden tagnak válaszolnia.

A SCRUM felépítése és működése

86

Lépések:



Sprint Review:

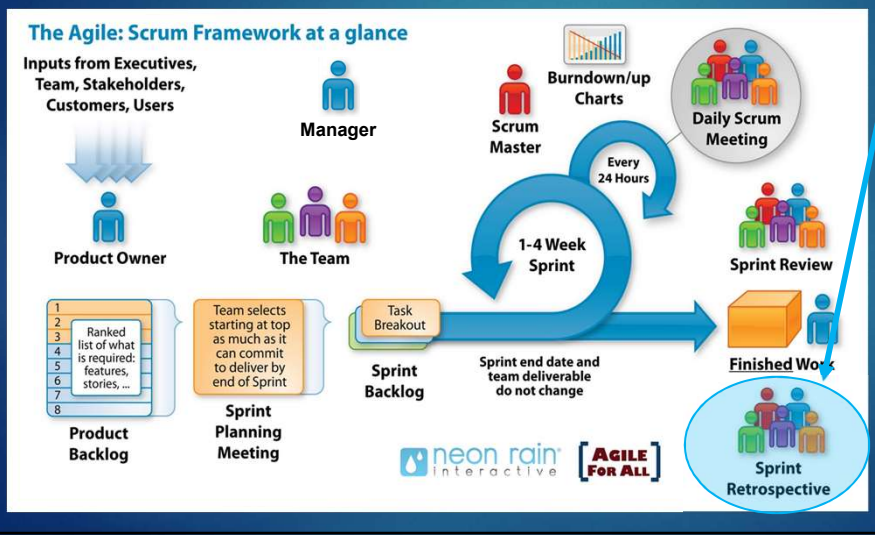
minden sprint végén van,

a sprint alatt befejezett sztorik értékelése,

a megrendelő is jelen van, ha elfogadják, a sztori készre jelenthető.

A SCRUM felépítése és működése

87



Lépések:

Sprint Retrospective:

Sprint Visszatekintés, felszínre kerülnek a munkát akadályozó tényezők.

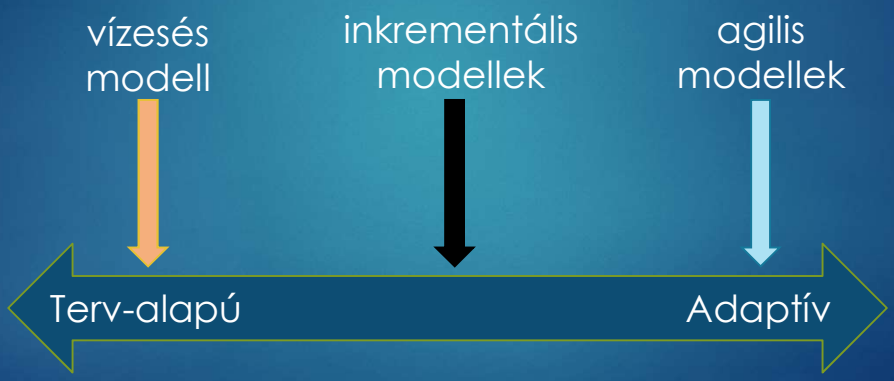
A SCRUM jellemzői

88

- Inkrementális, iteratív modell
- Az agilis módszertan szép példája
- Jól kezeli a megrendelő igényeinek változását
- Kommunikáció intenzív, sok a megbeszélés (félreértések tisztázása, vélemények ütköztetése)
- Hatékony csapatmunka, jól szervezett csapatok
- Adaptív menedzsment

2.8. Agilis vs. Tradicionális módszertanok

89

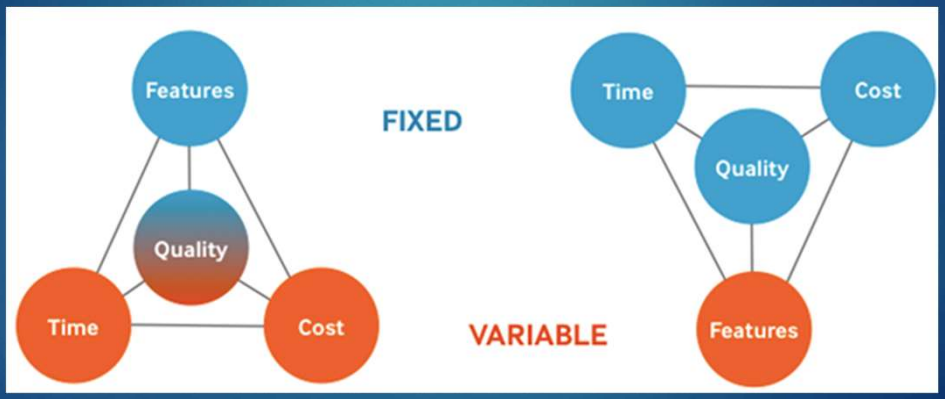


A projekt menedzsment „vas háromszöge”

90

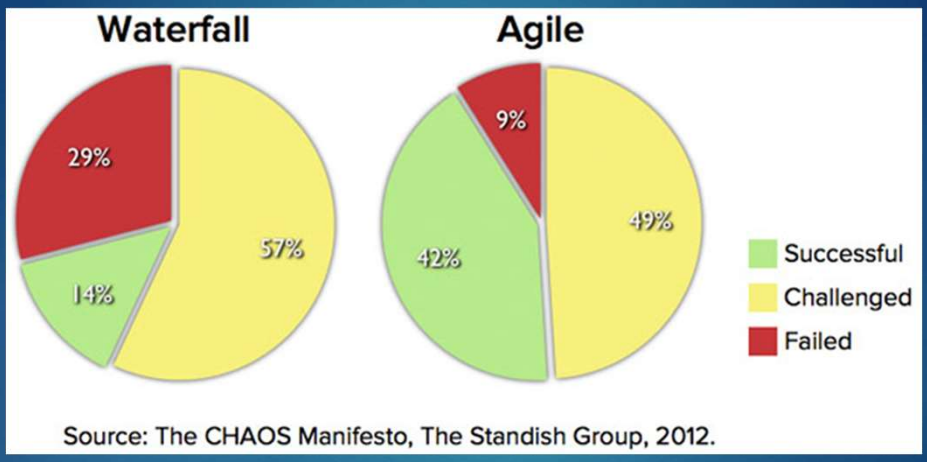
Tradicionális

Agilis



Az eddigi tapasztalatok alapján …

91



A hagyományos módszerek nem csak a vízesésből állnak ☺

3., A szoftverfejlesztés minőségi kérdései

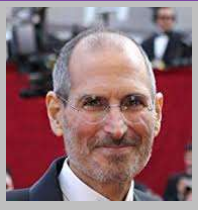
92

- Egyre komplexebb rendszerek → egyre égetőbb a minőség
- A minőség-menedzsment ma már elengedhetetlen része a szoftverfejlesztésnek (nincs minőségtanusítvány → piaci hátrány)
- Mi a minőség? Mi a szoftver minőség?
- A minőséget nehéz definiálni, lehetetlen mérni, de könnyű felismerni.

A minőség az, ha a vevő jön vissza és nem a termék.

„"We just can't ship junk. There are thresholds that we can't cross because of who we are." (Steve Jobs)

93



Steve Jobs
(1955-2011)
Informatikus, „vizionárius"
Az Apple, a NEXT, a PIXAR cégek alapítója, az Apple későbbi vezérigazgatója. Vezetése alatt fejlesztették ki az Apple I., II., a Macintosh, az iPod, az iPhone és az iPad termékeket.

3.1. A minőség megközelítése és definíciói

94

- Garvin a minőség fogalmának kutatása során 5 különböző megközelítésből indul ki a minőség definiálásához.
- Mind az 5 megközelítés arra keresi a választ, hogy mi van a minőség hátterében, mitől függ a termék minősége, mitől lesz egy termék vagy szolgáltatás jó minőségű.

David A. Garvin
(1952-2017)
A Harvard Business School „Business Administration" professzora
Minőséggel, vezetéselmélettel, menedzsmenttel, gazdaságtudománnyal, üzleti döntéselmélettel foglalkozott, ill. írt könyveket.

A minőség definíciói Garvin szerint

95

1. **Felhasználói alapú definíció:** A minőség a felhasználásra való alkalmasság. A terméknek a felhasználók egyedi igényeit kell a lehető legjobb minőségben kielégíteni. A legjobb termék az, amelyik ezt a legmagasabb színvonalon képes megtenni.
2. **Termék alapú definíció:** A minőség a termék mérhető paramétereiből következik. A termékek közti minőségi különbségek a termékek egyes összetevőinek vagy jellemzőinek a különbségeiből fakadnak. Ebben az esetben a minőség megítélése nem szubjektív, hanem objektív módon történik.

96

3. **Folyamat alapú definíció:** A minőség a specifikációnak való megfelelőséget jelenti.
4. **Érték alapú definíció:** A költség függvényében határozza meg a minőséget. A jó minőségű termék alacsony áron alkalmas a kitűzött feladat elvégzésére. A termék a specifikációnak elfogadható áron képes eleget tenni.
5. **Transzcendens definíció:** A minőség egyfajta veleszületett kiválóságot jelenti. Eszerint a minőség abszolút, egyetemes.

A minőség definíciója az ISO szerint

97

- Az egység (termék, szolgáltatás) azon jellemzőinek összessége, amelyek befolyásolják képességét, hogy meghatározott és elvárt igényeket elégítsen ki.
- A termék, vagy szolgáltatás azt nyújtja a vevőnek, amit akar, amikor akarja, és mindezt megfelelő árért.

A minőség dimenziói Garvin szerint

98

Garvin azt a kérdést vizsgálta, hogy a vevők számára a termék, vagy szolgáltatás mely tulajdonságaiból fakad a vevő elégedettsége?

- **Teljesítmény**, a termékek, szolgáltatások alapvető felhasználhatóságát jelenti.
- **Sajátosságok**, a termék alapvető jellemzője, amely megkülönbözteti más hasonló terméktől.
- **Megbízhatóság**, annak valószínűsége, hogy a termék a használati idő alatt nem romlik el.

99

- **Megfelelés,** követelmények kielégítése, melyek pl. szabványokban rögzítettek. Jelentősége a tömegtermelés megjelenésével nőtt, mert nincs mód a termékek egyedi ellenőrizésére (LEGO).
- **Tartósság**, a termék, szolgáltatás használati időn belül nyújtott teljesítményének mértékét jelenti.
- **Használhatóság**, a termék által kínált lehetőségek kiaknázhatóságának mértékét mutatja.
- **Esztétikai tulajdonságok**, a termék külső jegyeinek összessége.
- **Felismert minőség**, a márkanév, a termék jó híre a felhasználók körében tovább emeli a minőséget (Adidas, Gucci).

100

3.2. A minőség fokozatai

1., Az ártatlanság kora

- A vezetés szerint a minőségügy a gyártással együtt járó szükséges rossz
- A selejtet sorscsapásként érzékelik (utólag)
- A profit fontosabb, mint a vevő kívánsága
- A minőség-költségeket nem mérik
- A terméket „rásózzák" a vevőre („ez van, ezt kell szeretni")

101

- **2., Az eszmélés időszaka**

- A vezetés felismeri, hogy a minőség pénzbe kerül, de a fennmaradáshoz szükséges
- A vállalat struktúrájának átszervezése, problémák kiküszöbölése
- A minőség-ellenőrzés helyébe minőségbiztosítás lép (még kevés a hatásköre)
- Keresik a vevők véleményét

102

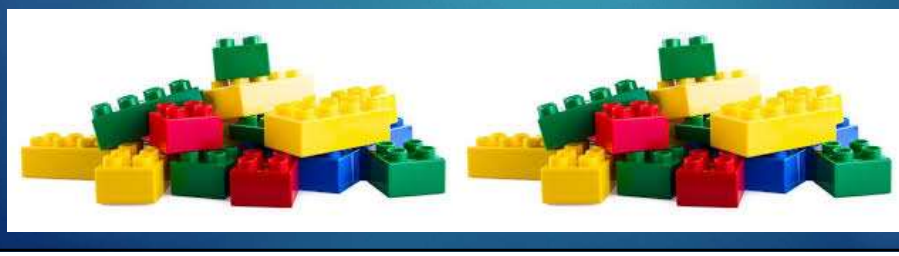
3., Az elkötelezettség kora

- A minőség gazdasági szükségszerűség
- A minőségbiztosítási rendszert megvalósítják és ellenőrzik
- A gyenge minőség kiküszöbölése 50% költség csökkenést eredményez
- A vevők véleményét kutatják, elemzik
- A munkások bevonása, nyereségmegosztás, továbbképzés és ellenőrzése

4., A világszínvonal elérése

103

- A minőségi szemlélet mindenek felett
- A megelőzés az élet útja
- Folyamatos, sosem befejezett tökéletesítés
- A vevők lelkesedése
- Minden munkatárs felelős a minőségért



3.3. A szoftverminőség megközelítései

104

A szoftverminőség megközelítései

Termék-alapú megközelítés

- McCall féle modell
- Boehm féle modell
- ISO 9126
- ISO 25010

Folyamat-alapú megközelítés

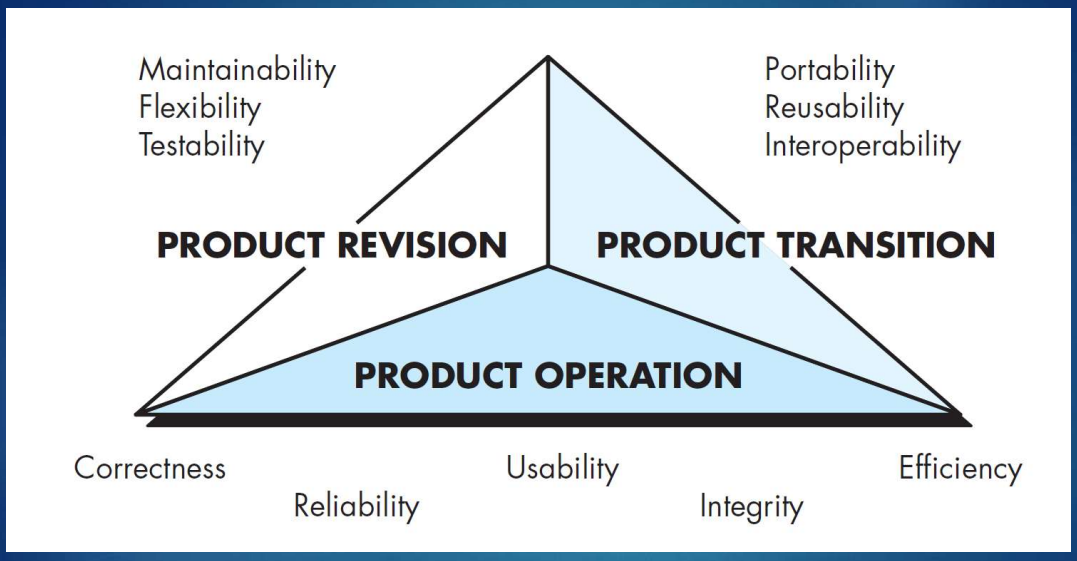
- ISO 9000

McCall féle minőségmodell

105

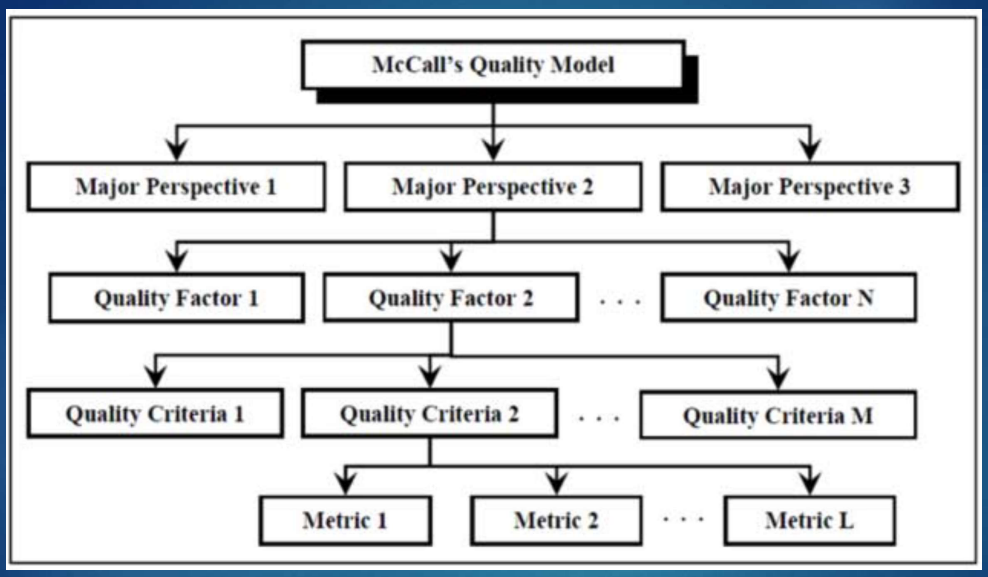
- McCall és munkatársai (General Electric) fejlesztették ki.
- A DoD, illetve az AirForce által kidolgozott termék alapú minőségmodellt vették alapul.
- A McCall féle modell a végterméket veszi alapul.
- A modellben hierarchikusan három szintet azonosítanak.
- A minőségi kritériumokhoz metrikákat rendelnek, melyek a termék jellemzőit határozzák meg.

106



A McCall féle minőségi háromszög

107



http://spring2012.stpcon.com/Item/1039/

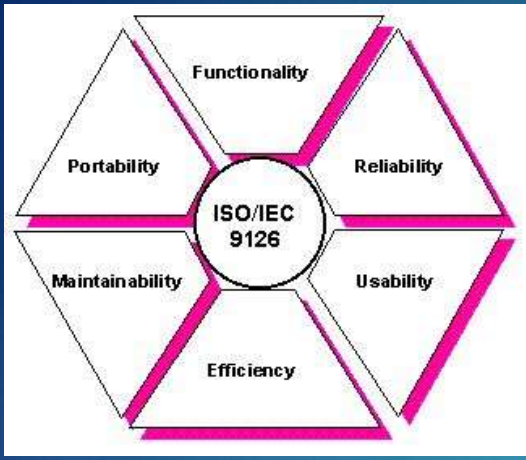
A McCall féle minőségi modell szintjei

Az ISO 9126 féle minőségmodell

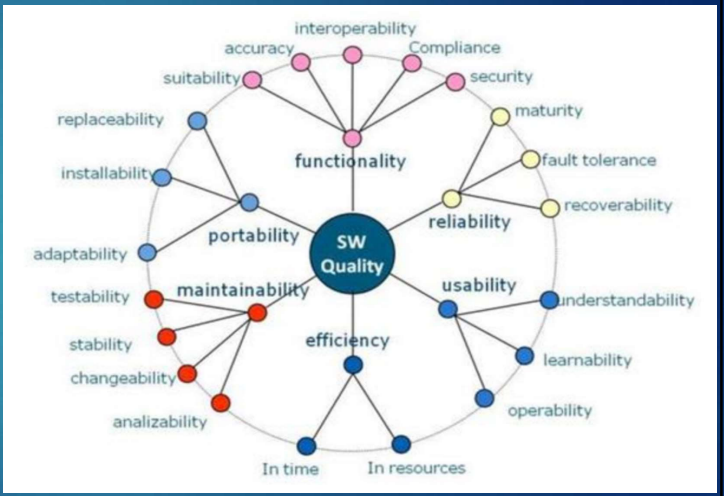
108

- Termék alapú, kvalitatív modell, a McCall és Boehm modelleket felhasználva alakult ki.
- A legrészletesebben fogalmazza meg a szoftvertermék minőségi attribútumait és mérőszámait.
- A szabvány felépítése:
  1. rész: Minőségi modell - ISO/IEC 9126-1:2001
  2. rész: Külső metrikák - ISO/IEC TR 9126-2:2003
  3. rész: Belső metrikák - ISO/IEC TR 9126-3:2003
  4. rész: Használat közbeni metrikák - ISO/IEC TR 9126-4:2004)

109

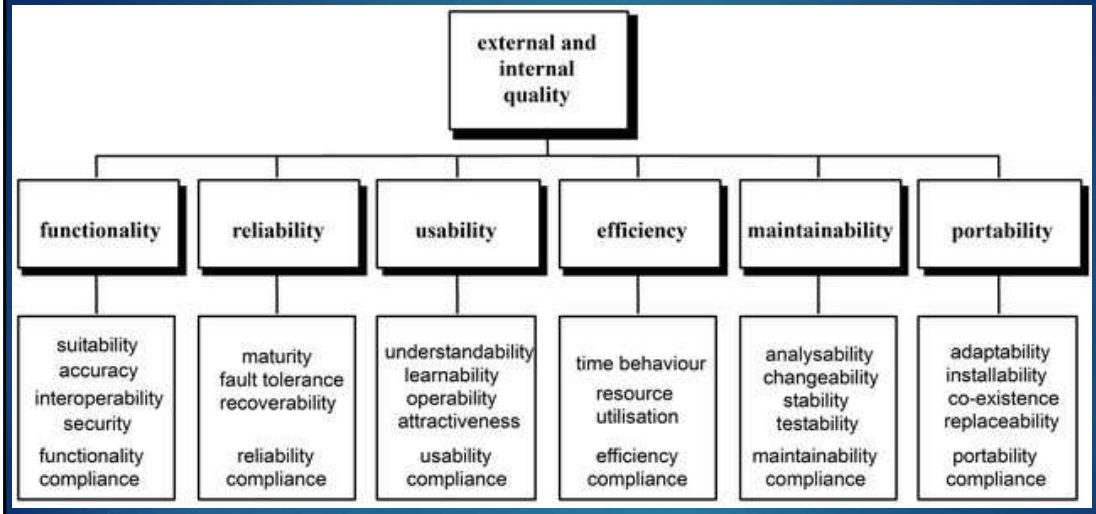


A 9126 minőségi jellemzői



A külső és belső minőségi jellemzők

110



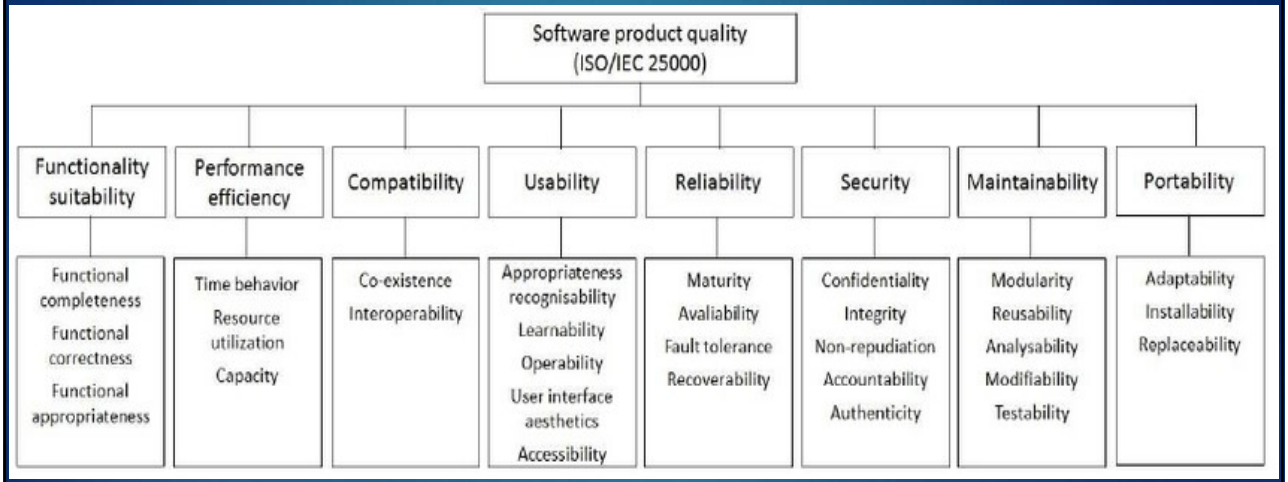
A 9126 külső és belső minőségi jellemzői

Az ISO 25010 féle minőségmodell

111

- A 9126 modernebb változata
- Teljes mértékben épít a 9126-ra
- Megjelenés 2011-ben, last review 2017-ben.
- legrészletesebben fogalmazza meg a szoftvertermék minőségi attribútumait és mérőszámait.
- A kor kihívásainak megfelelően 2 új elemmel bővült: Security, Compatibility
- Ezek eddig a Funcionality-n belül voltak

112



A 25010 minőségi jellemzői

A folyamat-alapú minőségmodellek

113

- Tömegtermelés → nem lehet minden terméket külön ellenőrizni.
- A termelési folyamat minősége meghatározza a termék minőségét.
- A folyamat termékét ellenőrizzük, eltérés esetén állítunk a folyamaton, ez visszahat a termék minőségére.
- A jól beállított folyamat hibahatáron belüli terméket eredményez (ehhez többet nem nyúlunk).

A szoftver termék esetében

114

- Az előállítási folyamat és a termék minősége közötti kapcsolat bonyolultabb.
- Egyes lényeges jellemzők nem mérhetők anélkül, hogy a szoftvert huzamosabb ideig ne használtuk volna (pl. karbantarthatóság).
- A tervezés, mint folyamat minőségi szempontból nehezen tartható kézben (kreativitás hatása).
- Gyakorlati tapasztalat: a folyamat minősége alapvető hatással van a termék minőségére.
- A folyamat minőségkezelésének, továbbfejlesztésének van létjogosultsága a szoftver esetében.

Az ISO 9000-es szabvány

115

- Célja a fejlesztési folyamat értékelése, hitelesítése és minőségének rögzítése
- Az ISO 9000-es szabványcsalád foglalkozik ezzel a területtel
- Az ISO 9000-3 szabvány: a szoftver fejlesztés specialitásait veszi figyelembe

116

- Nem garantálja a termék piaci sikerességét!
- Egy termék gyártásának ISO minősítése nem a termék kiválóságát jelenti, hanem azt, hogy a gyártás minőségbiztosítási rendszere az ISO 9000 szerint lett kidolgozva.
- A legelterjedtebb szabvány.

Az ISO 9000 minőségirányítási alapelvei

117

- A szabvány nyolc minőségirányítási alapelvet határoz meg.
- Ezek a szervezetek vezetésének és működtetésének alapvető szabályai.
- Cél: a szervezet teljesítményének fokozatos fejlesztése, a vevőkre való összpontosítás, a többi érdekelt fél igényeinek figyelembe vétele,
- Ezek a minőségirányítási alapelvek, beépültek az ISO 9000-es szabvány-sorozat tartalmába.

118

1. **Vevőközpontúság:** A szervezetek a vevőiktől függenek, ezért meg kell érteniük a jelenlegi és jövőbeli vevői igényeket, teljesíteniük kell a vevők követelményeit és törekedniük kell a vevői elvárások túlteljesítésére.

2. **Támogató vezetés:** A vezetők hozzák létre a szervezet céljának és irányításának egységét és azt a belső környezetet, amelyben a munkatársakat teljes mértékig be lehet vonni a szervezet céljainak elérésébe.

119

3. **A munkatársak bevonása:** A szervezet lényegét minden szinten a munkatársak jelentik, és teljes bevonásuk teszi lehetővé képességeik felhasználását a szervezet javára.

4. **Folyamatszemléletű megközelítés:** A kívánt célt hatékonyabban lehet elérni, ha a tevékenységeket és a hozzá kapcsolódó erőforrásokat folyamatként kezelik.

120

5. **Rendszerszemlélet az irányításban:** Az egymással összefüggő folyamatok rendszerként való azonosítása, megértése és irányítása hozzájárul ahhoz, hogy a szervezet eredményesen és hatékonyan valósítsa meg céljait.

6. **Folyamatos fejlesztés:** A szervezet állandó célja legyen az átfogó, teljes működésre vonatkozó folyamatos fejlesztés.

121

7. **Tényeken alapuló döntéshozatal:** Az eredményes döntések az adatok és egyéb információk elemzésén alapulnak.
8. **Kölcsönösen előnyös kapcsolatok a beszállítókkal:** A szervezetek és beszállítóik kölcsönösen függnek egymástól, értékteremtő képességük a kölcsönösen előnyös kapcsolatok révén növekszik.

122

Az ISO 9000 és a minőségkezelés

4., A szoftver fejlesztés biztonsági kérdései – a biztonságos kód

123

- „Stand alone" rendszerek → a szoftverhiba, biztonsági rés helyi hatású.
- „Hálózatosítás" → a hiba kiterjedhet az intézmény hálózatában lévő eszközökre.
- „Internetesítés" a hiba bárhonnan, bármikor kihasználható.
- A KGB visszahozta a írógépeket ...

A Stuxnet „story"

124

- Speciális, igen komplex vírus, amely 3 rétegen keresztül működik Windows oprendszer, a Siemens Simatic folyamatfelügyeleti rendszer, PLC[3] programok.
- Kifejezetten az iráni urándúsító centrifugáinak inverteres motormeghajtóira szállt rá a féreg.
- 2010. november 16-án Irán leállította az urándúsítóit, miután a centrifugák több, mint 20%-a megsemmisült a Stuxnet tevékenysége nyomán, azaz a kártevő elérte a célját.
- Azóta az első cyber-fegyverként tartják számon
- Nagy team, szoftver tervezés, sok pénz kellett hozzá, ma sem tudni pontosan, hogy kik írták …

Biztonsági szempontból nézve a támadás tárgya lehet: 125

- Szoftverhiba, ténylegesen hibásan működő funkcionalitás (a komplexitás miatt előfordul,)
- Hibás inputszűrés (az első védelmi rendszer hibája)
- Feledékenységből „nyitva hagyott ajtó" (több feltétel együttállását nem vizsgáljuk, nem szűrjük ki).
- Szándékosan „nyitva hagyott hátsóajtó" (későbbi illegális belépésre, vagy magunknak vészmegoldásként felhasználva).

Kódolási hibák száma átlagosan 126

- Pontos adat nincsen, (a cégek „szégyenlősek") az alábbi közelítő adatok érhetők el:
- Ipari átlag: 15-50 hiba / 1000 sor, befolyásoló tényezők: a fejlesztési módszertan, a tesztelés minősége, a fejlesztő képességei, stb.
- Microsoft termékek: 10-20 hiba / 1000 sor a belső tesztelés előtt, 0.5 hiba / 1000 sor a kiadás után
- Legjobb módszerek: pl.: Harlan Mills: „cleanroom development": 3 hiba / 1000 sor tesztelés előtt, 0.1 hiba / 1000 sor kiadás után

4.1. A szoftverhibák osztályozása

127

- A legjobb módszertannal és teszteléssel is maradnak hibák a szoftverekben, amelyek utólag derülnek ki.
- A hibák biztonsági szempontból különböző hatásúak lehetnek. A hibákat hatásuk alapján a következő csoportokba sorolhatjuk:

1. **Biztonsági hatásuk nincs,** csupán a felhasználót idegesíti a hiba, inkább funkcionális hiba, mint biztonsági

128

2. **Alacsony hatású biztonsági hiba:** nem igényel azonnali beavatkozást pl. információszivárgás (beszédes hibaüzenet: osztálynév vagy adatbázis táblanév kiszivárog), biztonsági szempontból önállóan és közvetlenül nem jelent kockázatot

3. **Közepes hatású biztonsági hiba:** közvetlen biztonsági kockázatot jelent és sürgős beavatkozást igényel pl. Cross-site scripting sérülékenység (megtévesztéssel együtt alkalmazva kliensoldali adatok szerezhetők meg webbrowser-en keresztül) (Alkotmánybíróság weboldalán így írták át az alaptörvényt)

129

4. **Magas hatású biztonsági hiba:** azonnali beavatkozást igényel, pl. SQL injection (a támadó a felhasználók bizalmas adataihoz férhet hozzá a szerveren a szerver oldali kód hibája miatt)

5. **Kritikus biztonsági hiba:** pl. Buffer overflow az Operációs rendszer valamelyik api-jában (a hibán keresztül távolról kód futtatható az operációs rendszeren, a sérülékeny szoftvert emberek milliói használják naponta)

Szoftverhibák helye

130

- Egyedi szoftverekben vagy weboldalakon: A hibás szoftverkód egy egyedi fejlesztés eredménye, a hibás kód máshol nem fordul elő. (pl. hibás php script egy egyedi weboldalon)
- Tömegtermékekben: A hibás szoftverkód egy hivatalosan kiadott szoftvertermékben van jelen pl. Weboldalak CMS szoftverében, Webszerverek kódjában, Asztali alkalmazásokban, Mobile app-okban, stb.

Sérülékenységi adatbázisok

Sérülékenységi adatbázisok

131

- A tömegesen használt és hivatalosan kiadott termékekben lévő hibákat nyilvántartják és folyamatosan regisztrálják:
- Computer Emergency Response Team (CERT), 1988-ban alapította a DARPA (az Egyesült Államok védelmi minisztériumának kutatási ügynöksége) 1988 óta közel félmillió biztonsági incidensről érkezett bejelentés.

US-CERT
UNITED STATES COMPUTER EMERGENCY READINESS TEAM

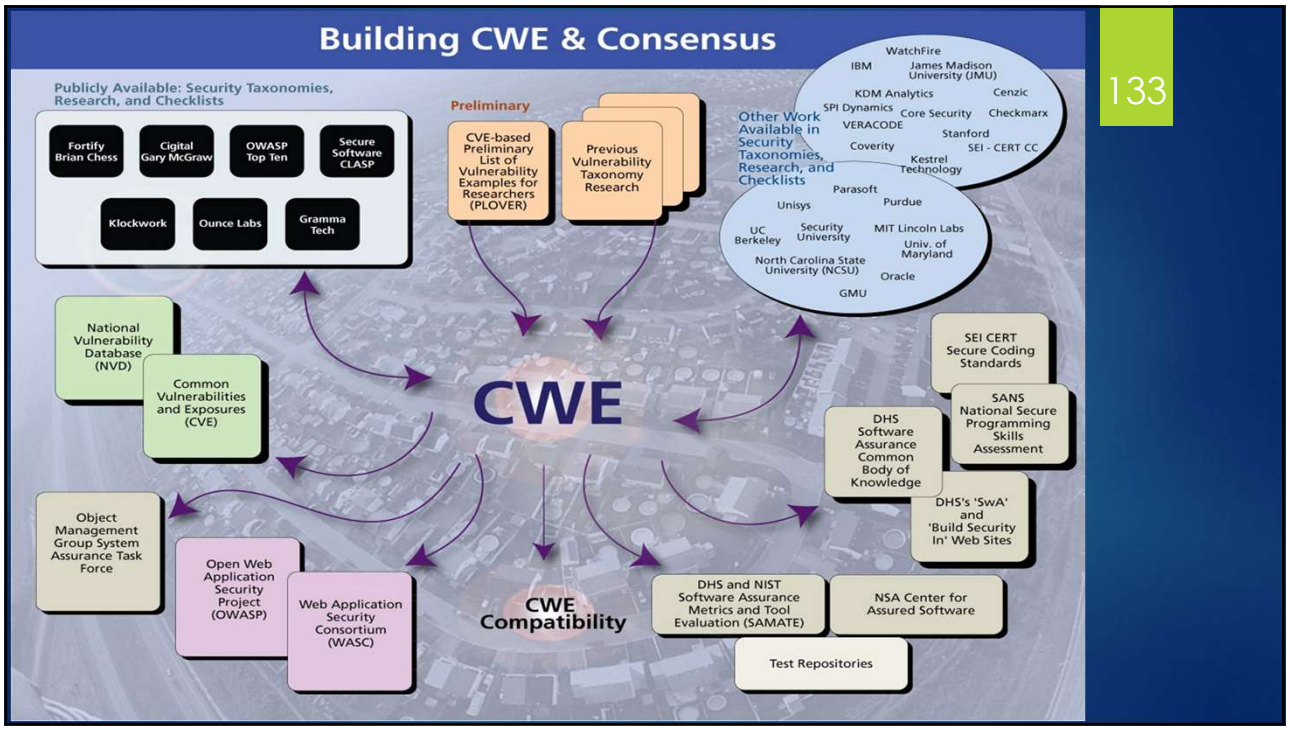
GOV CERT MAGYARORSZÁG NBSZ

Kormányzati Eseménykezelő Központ
www.cert-hungary.gov

132

- **Common Vulnerabilities and Exposures (CVE)**, Minden egyes sérülékenységet egy szabványos számmal lát el pl.: CVE-2007-0038: a kernel32.dll LoadAnilcon metódusában lévő kritikus WinXP buffer overflow hiba. (www.cve.mitre.org)
- **Common Weakness Enumeration (CWE)**, Nemzetközi nyílt egységesített gyűjteménye a szoftver sérülékenységeknek. Igyekszik hatékonyan támogatni a sérülékenységeket elkerülő architektúrákat és tervezést. (www.cwe.mitre.org)

CWE Common Weakness Enumeration
A Community-Developed Dictionary of Software Weakness Types



4.2. Sérülékenység fajták a CVE adatbázis alapján

134

- Szolgáltatás megtagadás (DoS, Slow-Dos, Dirtibuted DoS)
- Kódvégrehajtás (saját kód futtatása, kalkulátor megnyitása)
- Túlcsordulás
- Cross Site Scripting (XSS)
- Könyvtár átjárás
- Valami megkerülése
- Információ szerzés
- Jogosultság szerzés
- SQL injekció
- Fájl beszúrás
- Memória korrupció
- Cross Site Request Forgery (CSRF)
- HTTP válasz szétosztás

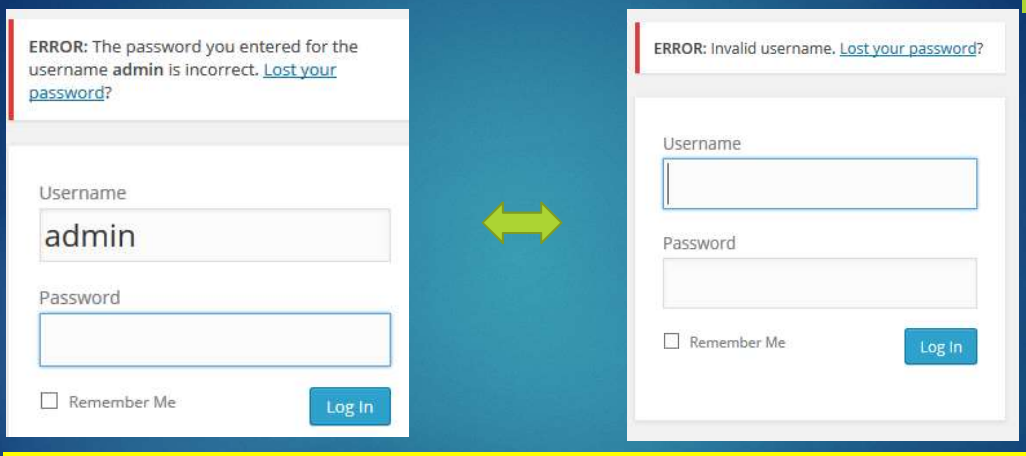
Információ szerzés / Információ szivárgás

135

- A szoftver a lehető legkevesebb adatot szolgáltassa! A felhasználót nem érintő (felesleges) adat ne kerüljön ki!
- Gyakori példák:
  - szolgáltatás, operációs rendszer verziószám kiszivárgás
  - belső fájlszerkezet kitakarása
  - beszédes hibaüzenet belépő felületnél
  - beszédes hibaüzenet hibás input adatra
  - belső kommentek kiszivárgása
  - stb.

Példa: felhasználói névre vonatkozó információ a wordpress-ben

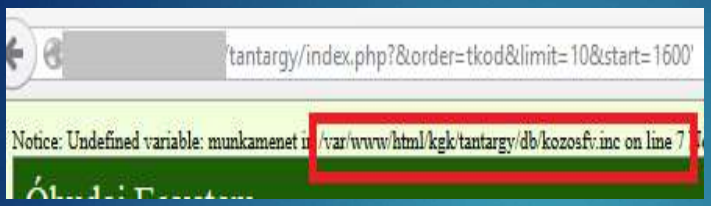
136



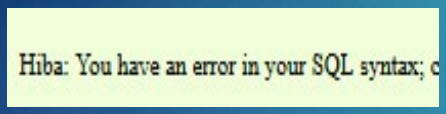
A hibaüzenetek csak a minimális információt tartalmazzák!
(nem kell tudni, hogy van admin user)
Tipikus próbálkozás: admin/admin

Példa: beszédes, „fecsegő” hibaüzenetek

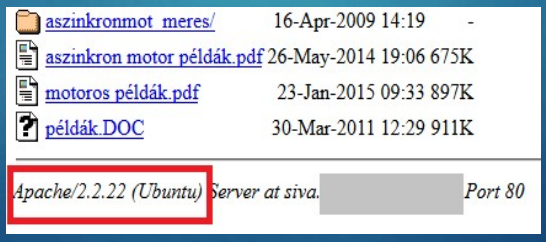
137



Webroot kitakarása



Webszerver verzió kitakarás



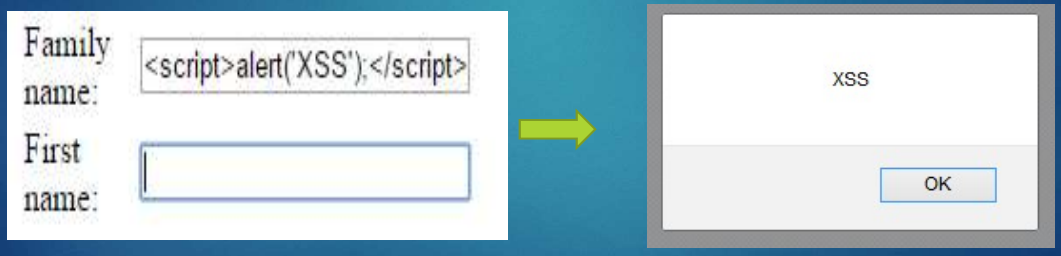
Cross Site Scripting (XSS)

138

Miért nem jó az alábbi php kód?

```
<?php if (isset($_POST["vnev"])) { print("Üdvözlöm ".$_POST["vnev"]."!"); } ?>
```

A script bármely input adatot elfogad ellenőrzés nélkül, input validáció mindig szükséges!



A hibát kiaknázva kliens oldali kódot futtathat a böngésző !

Cross Site Request Forgery

139

Minden művelet előtt jogosultság ellenőrzésnek kell lennie, amennyiben az ügyfélnek nincs session-je, az alábbi művelet nem hajtódik végre:
http://bank.com/transfer.do?acct=BOB&amount=100

A támadó szétküldi az alábbi linket az áldozatainak:
http://bank.com/transfer.do?acct=ATTACKER&amount=100000

Azok az áldozatok, akik éppen be voltak jelentkezve a banki rendszerben (van aktív session) végrehajtják az utalást!

SQL Injekció

140

Miért nem jó az alábbi php kód?

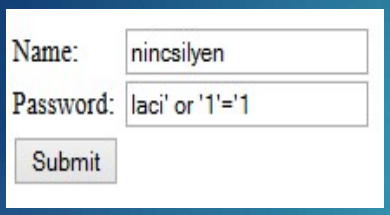
```
$connect = @mysql_connect ($host, $user, $pass);
@mysql_select_db($database,$connect) or die( "Unable to select database");
if ( $connect ) { $result = mysql_query("SELECT * FROM Tabla1 Where email="".$_GET["email"]."""); }
```

Hiányzik az input validáció, a script bármely input adatot ellenőrzés nélkül betesz az SQL lekérdezésbe!

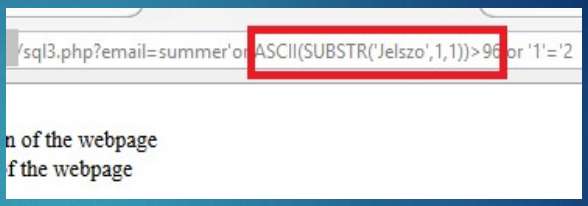
Példa SQL Injekcióra

141

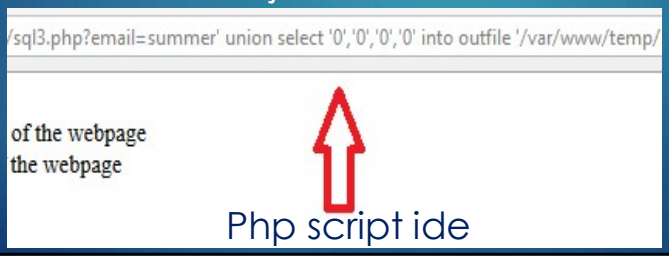
bejelentkezés



Igaz/hamis lekérdezés



Fájl feltöltés



Memória korrupció

142

Stack túlcsordulás:

```
#include <string.h>
void func1(char* ar1)
{ char ar2[10];strcpy(ar2,ar1); }

int main(int argc, char* argv[])
{   func1(argv[1]); return 0; }
```

Amennyiben az input adat nagyobb mint 10 karakter, a func1 metódus visszatérési címe a stacken felülíródik!

Példák memória korrupcióra

143

- veszélyes olyan metódusok használata, amelyek nem végeznek hossz validációt pl. strcpy, gets, puts, stb…
- tömbmásolás méretellenőrzés nélkül
- túl kevés paraméter vagy ellenőrzés nélküli input adat használat formázott stringekben, pl.: printf(„valami %u valami"); a %u miatt egy változónak kellene lennie a string után: printf(„valami %u valami",x);
- objektumok használata a felszabadításuk után:
  free(a); a->b = 5; (Use after free)
- objektumok kétszeri felszabadítása (Double free):
  if (feltétel1) free(a); if (feltétel2) free(a);

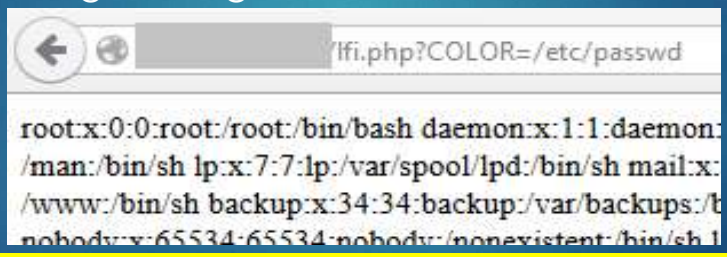
Könyvtár átjárás

144

Miért nem jó az alábbi php kód?

<?php include($_GET["COLOR"]; } ?>

A script bármely input adatot elfogad ellenőrzés nélkül, input validáció mindig szükséges!



A bemeneti adat ellenőrzésének hiánya miatt könyvtár átjárás lehetséges!

Fájl beszúrás

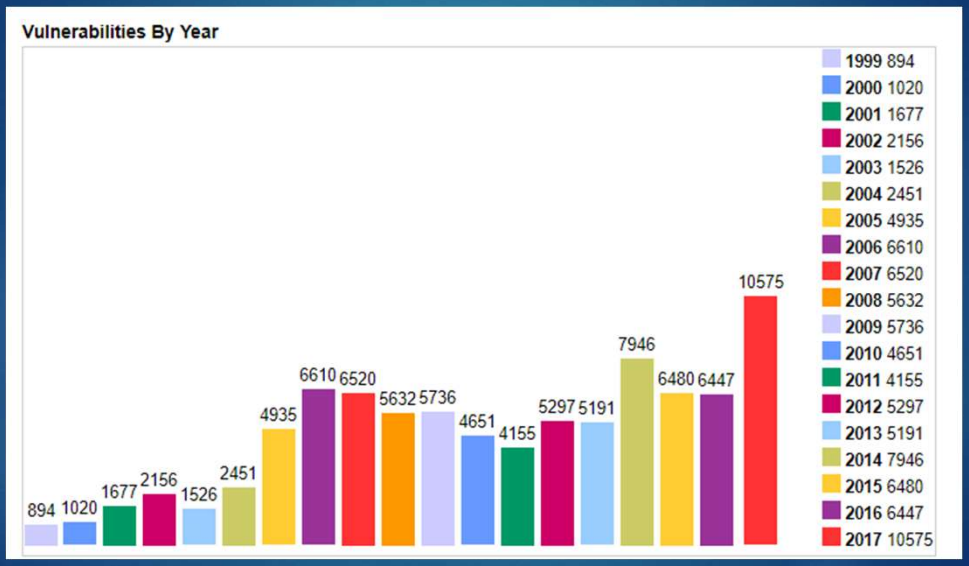
145

Fájlfeltöltés nem megfelelő ellenőrzés miatt, pl:

- futtatható fájlok feltöltése ellenőrzés nélkül
- futtatható fájlok feltöltése más formátummal: tamadokod.jpg
- kettős kiterjesztés: tamadokod.jpg.php
- fájlfeltöltés sql utasítással (insert into outfile)
- Stb.

Sérülékenységek megoszlása évek szerint

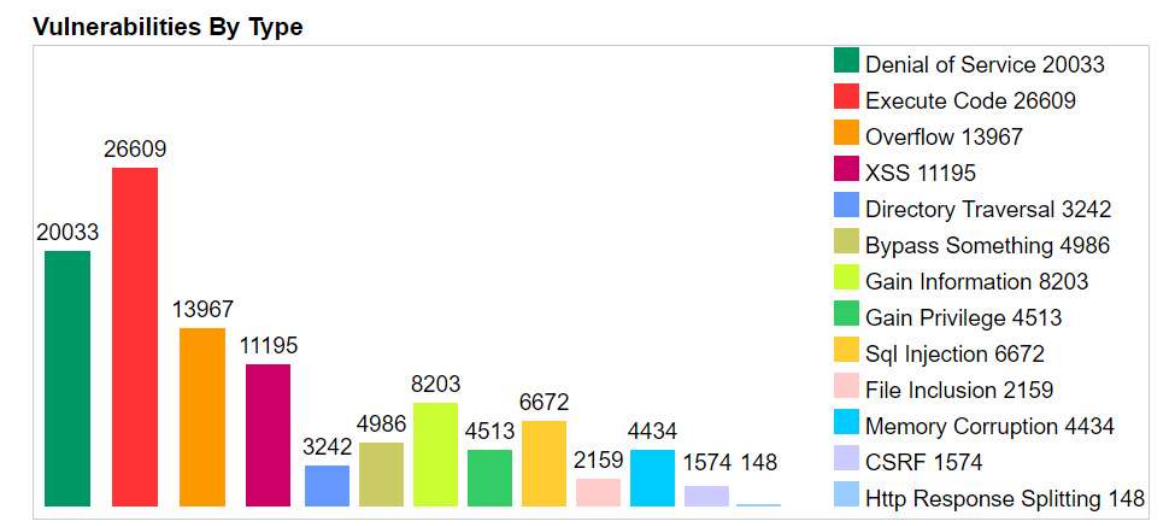
146



Forrás: https://www.cvedetails.com/browse-by-date.php

Sérülékenységek megoszlása típus szerint

147



Forrás: https://www.cvedetails.com/browse-by-types.php

4.3. A Microsoft SDL módszertan

148

- A Microsoft Security Development Lifecycle egy speciális szoftverfejlesztési folyamat.
- Jellemzője, hogy növeli a szoftverek megbízhatóságát, különös tekintettel a biztonsági hibákra.
- 2004. januárjában készült el az első verzió, jelenleg a 2012-ben kiadott 5.2-es verzió érvényes

Biztonsági koncepciók

149

Alapvető

- Tervezés a biztonságra
- Támadási felület csökkentése
- Mélységi védekezés
- Legkisebb privilégium elve
- Biztonságos defult beállítások

További

- Biztonságos architektúrák
- Felhasználó felület tervezés
- Részletes biztonsági megfontolások
- Válaszadás biztonsági esetekre
- Speciális veszélycsökkentő eljárások implementálása

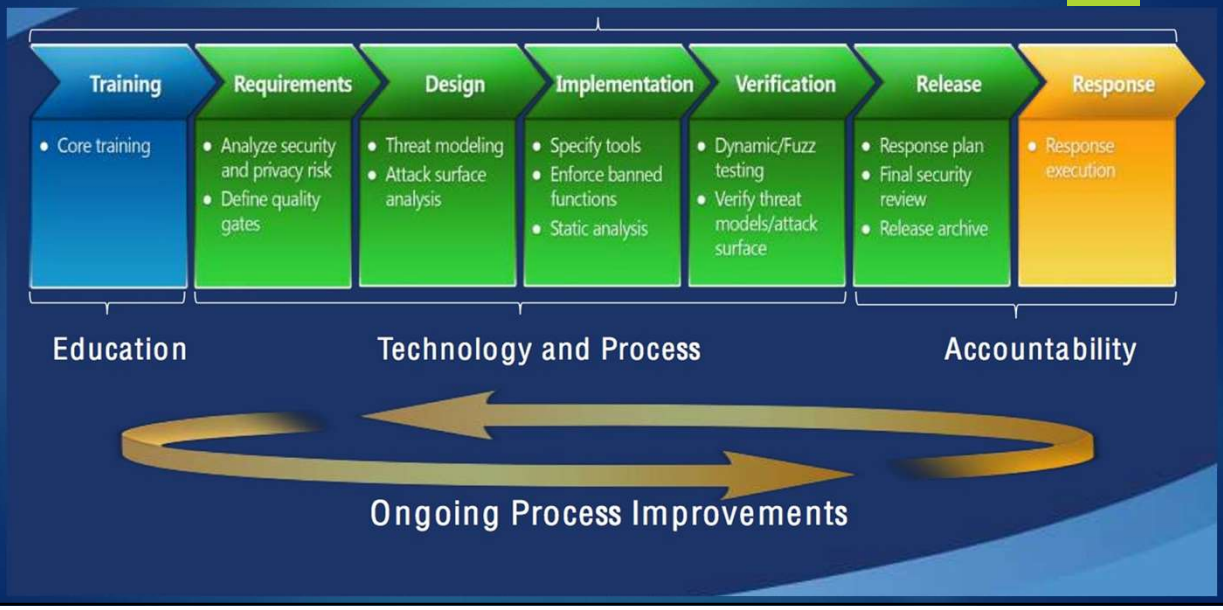
A Microsoft SDL módszertan lépései

150

0. szakasz: Oktatás (biztonsági problémák, megoldások ismerete)
1. szakasz: Követelményspecifikáció
2. szakasz: Tervezés
3. szakasz: Implementáció
4. szakasz: Verifikáció
5. szakasz: Kiadás
6. szakasz: Utóélet

A Microsoft SDL életciklus modellje

151



0. szakasz: Oktatás

152

- A szoftverfejlesztésben résztvevő minden kolléga részvétele (fejlesztő, tesztelő, manager)
- Biztonsági problémák alapjai
- Legújabb biztonsági problémák
- Legújabb megoldások

1. szakasz: Követelményspecifikáció

153

- Biztonsági követelmények
- Bizalmassági követelmények
- Biztonsági ajánlások
- Erőforrások

Biztonsági követelmények

- Szükséges-e SDL
- Biztonsági kérdések követésében felelős személy kijelölése
- Bug reporting – biztonsági problémák is kerüljenek bele
- Minimális szint kijelölése biztonság szempontjából

154

Bizalmassági követelmények:

- Bizalmas adatok kezeléséért felelős kontakt személy
- Bizalmas adatokkal kapcsolatos kérdések dokumentálása (privacy bug bar)

Biztonsági ajánlások:

- Oktatás
- Fenyegetés modellezés
- Végső biztonsági review

Security bug effektus (STRIDE módszer)

155

Security threats:

Spoofing :	átverés
Tampering:	babrálás
Repudiation:	megtagadás
Information disclosure:	információ szivárgás
Denial of service:	szolgáltatás megtagadása
Elevation of privilege:	jogosultság emelés

A security bug okai

156

- Buffer overflow (puffer túlcsordulás)
- Aritmetikai hiba
- Sql/script injection
- Directory traversal (könyvtár átjárás)
- Race condition
- XSS
- Kriptográfiai hiba
- Authentikációs hiba
- Authorizációs hiba
- Korlátlan erőforrás foglalás
- Hibaüzenet hibája
- Egyéb

2. szakasz: Tervezés

157

- Biztonságos architektúra
- Támadási felület minimalizálása
- Legkisebb jogosultsági szintek
- A default installáció biztonságos legyen
- Mélységi védekezés
- Új verzió esetén régi hibáinak figyelembe vétele
- Nem szükséges funkciók kivétele
- Hibaüzenetek csak a szükséges adatokat adják meg
- Ahol lehet menedzselt kódot kell használni
- Session menedzsment

3. szakasz: Implementáció

158

- Minimális kódgenerálás
- Kód review eszközök használata
- Tiltott string kezelő eljárások kerülése
- Input validáció, output encoding
- Ad-hoc sql query-k kerülése
- Web szervizek kezelése megfelelő xml parse-lel
- /GS flag használata

(folytatás)

159

- ASLR használata (Address Space Layout Randomization) engedélyezni kell minden nativ kód esetében.
- Biztonságos adatbáziselérés
- LINQ ExecuteQuery kerülése
- Exec kerülése a tárolt eljárásokban
- Biztonságos cookie-k használata
- Biztonságos dll betöltés
- Biztonságos redirect

4. szakasz: Verifikáció

160

- Fuzzerek használata (RPC fuzzer, ActiveX fuzzer, stb.)
- Hibák dokumentálása
- Webszervízeknél xml parser teszt
- Memória korrupciós tesztek
- Hálózati fuzzolás
- Bináris vizsgálata

Tesztelés

161

- Teszttterv készítése
- Penetrációs teszt
- Regressziós sérülékenység teszt
- Authentikációs tesztek
- Visszajátszásos tesztek
- Kód review

5. szakasz: Kiadás

162

- Bizalmasság review
- Biztonsági felelős kijelölése a kiadás utánra
- Eljárás kidolgozása biztonsági problémákra
- Trace és debug letiltása
- Végső biztonsági review

Az EVOSOFT által tartott előadások témái

163

- Konfiguráció kezelés, folyamatos integráció
- Projekt átadás és rendszerintegráció
- Domain modell
- Szisztematikus Architektúra Tervezés
- Rizikó alapú tesztelés
- Tesztelés az ipari gyakorlatban