

VÁLLALATI INFORMÁCIÓS RENDSZEREK

előadás

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI  2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

VERZIÓKÖVETÉS FOGALMA ÉS ESZKÖZEI

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

VERZIÓKÖVETÉS CÉLJA

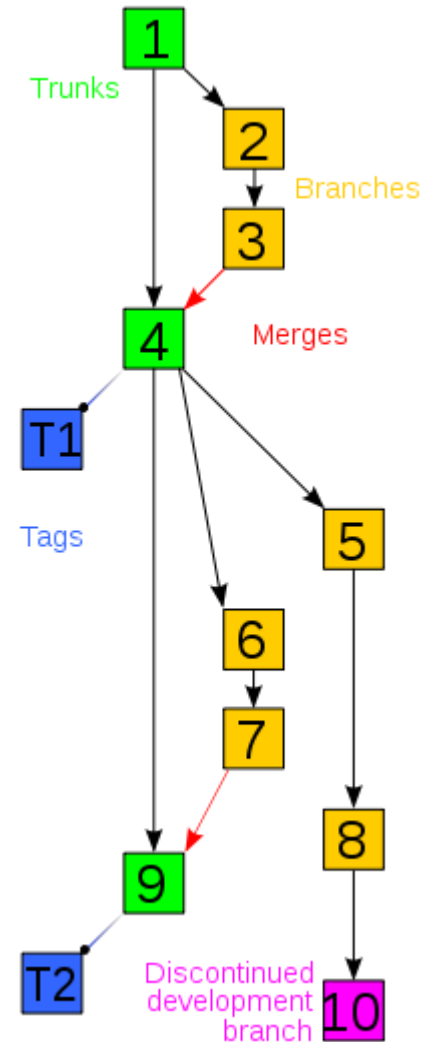
- Forráskód és egyéb szöveges (vagy bináris) állományok (asset) változásainak nyomon követése
 - Minden változtatás egyedi azonosítóval ellátott (revision)
 - Változás meta-információi
 - Ki, mikor, mit módosított
 - Megváltozott sorok (diff)
- A változások visszakereshetők, visszaállíthatók
- Különböző változások összefésülhetők (merge)
- Csapatok által fejlesztett közös kódbázis esetén elengedhetetlen

VERZIÓKÖVETÉS MEGVALÓSÍTÁSA

- Speciális szoftverek (VCS – Version Control System)
 - Számos különböző koncepció a fenti célok megvalósítására
- Változások tárolása
 - Fájlok/adatok kezdeti változatának tárolása, módosítások nyomon követése
 - Intuitív, de átnevezés, törlés, összefésülés problémás
 - Minden változás az egész adathalmaz egy újabb állapota
 - Apró/egyszerű módosítások esetén kevésbé intuitív
 - Nagy/komplex változtatások esetén viszont egyszerűsít

VÁLTOZÁSOK SZERKEZETE

- Az egyes verziók (revision) irányított gráfként reprezentálhatók
 - Directed Acyclic Graph (DAG)
 - Gráf pontjai az egyes revíziók
 - Élek a rákövetkező revízió kapcsolat
 - Több párhuzamos fejlesztési ág
 - Tipikusan van egy fő ág
 - Az összefűzés (merge) műveletek miatt nem fa
 - A változások szekvenciálisan követik egymást időben
 - Sorba rendezhető revízió szám vagy időbélyegző alapján
 - HEAD jelöli a legutolsó revíziót



FEJLESZTÉSI ÁGAK

- Triviális esetben egy lineáris fejlesztési ág (branch) létezik
 - Minden revíziót pontosan egy másik revízió előzi meg
 - A verziók egyszerű vonalat képeznek
 - Egyetlen legfrissebb verzió
 - Ez a HEAD
- Tipikusan több párhuzamos fejlesztési ág létezhet
 - Több későbbi revízió is követhet egy korábbi revíziót
 - Minden ágnak van legfrissebb verziója
 - Gyerek nélküli revíziók

TÖBB ÁGAS FEJLESZTÉS

- Mivel több „legfrissebb” revízió van, HEAD nem egyértelmű (vagyis minden ágon van egy)
 - Tipikusan kineveznek egy preferált HEAD-et
 - Általában ez a fő fejlesztési ág HEAD revíziója
 - Ez lesz a fő HEAD revízió
 - Minden HEAD-en alapuló későbbi revízió vagy az új HEAD, vagy egy új ág lesz
- Mindig létezik egy kitüntetett fejlesztési ág, ez a főág
 - Különböző elnevezése lehet: *trunk*, *mainline*, *master*, *base*, stb.
 - A kiindulási revíziótól a HEAD-ig tartó verziók listája
 - A revízió gráfon megtett legrövidebb séta, ami egy lineáris részgráfot képez
- Amikor egy revízió több korábbi verzión alapul
 - Összefűzés (merge) művelet
 - Két vagy több ág módosításainak egy közös ágba olvasztása

EGY VALÓS HISTORY GRÁF...



Forrás: <http://endoflineblog.com/assets/gitflow-mess-cfd9aa7a4137e3e575510fcbbf38a5b6.png>

ALAPVETŐ VCS MŰVELETEK

- Repository létrehozása/inicializálása
 - Import
- Repository tartalmának letöltése első alkalommal
 - Checkout
 - Clone
- Fájl módosítások feltöltése
 - Commit/Check-in
 - Push
- Lokális fájlok szinkronizálása a repository-val
 - Update
 - Fetch/Pull
- Módosítás történet lekérése (log)
- Fájl verziók közti különbség megtekintése (diff)

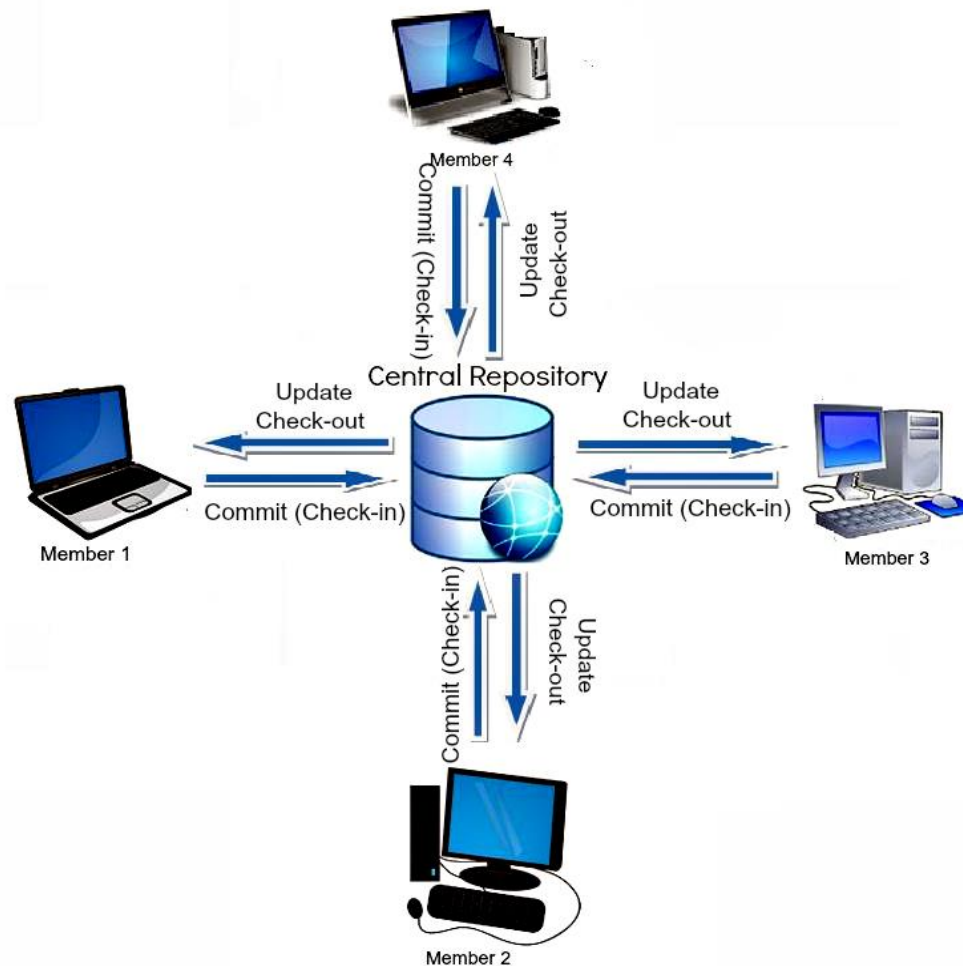
VCS MEGVALÓSÍTÁSI STRATÉGIÁK

- Két uralkodó stratégia VCS megvalósítására
 - Központosított (centralized) megvalósítás
 - Kliens-szerver architektúra
 - Repository, history, stb. egy központi szerver komponens által kerül tárolásra
 - Fájlok párhuzamos szerkesztését a szerver kezeli
 - Fájlok kezelése, kommunikáció a szerverrel kliens programok segítségével történik
 - Elosztott (distributed/decentralized) megvalósítás
 - Peer-to-peer megközelítés
 - Nincs központi repository, minden peer „saját” repository-val rendelkezik
 - Nincs központi szinkronizáció
 - Peer-ek egymás között szinkronizálhatják a módosításokat

KLIENS-SZERVER MEGVALÓSÍTÁS

- Központi szinkronizáció a szerver által
 - Több kliens által párhuzamosan szerkesztett fájl gondot okoz
 - Megoldás 1: file zárolás (lock) segítségével
 - Miután a kliens letölti a fájlt szerkesztésre, az zárolódik, és mások által nem módosítható a zárolás feloldásáig
 - Viszonylag biztos módszer, de a hosszú zárolás miatt nem hatékony
 - Megoldás 2: a verziók összefűzésével (merge)
 - A fájlok párhuzamosan szerkeszthetők
 - Az első módosítás biztosan megtörténik
 - Minden ezt követő módosítást összefűznek az előzőkkel
 - Hogy a korábbi módosítások ne íródjanak felül
 - Lehet automatikus, fél-automatikus
 - Nem mindig triviális, és végezhető el kézi beavatkozás nélkül
 - Conflict!!!
 - Példák: Subversion, CVS, ClearCase, stb.

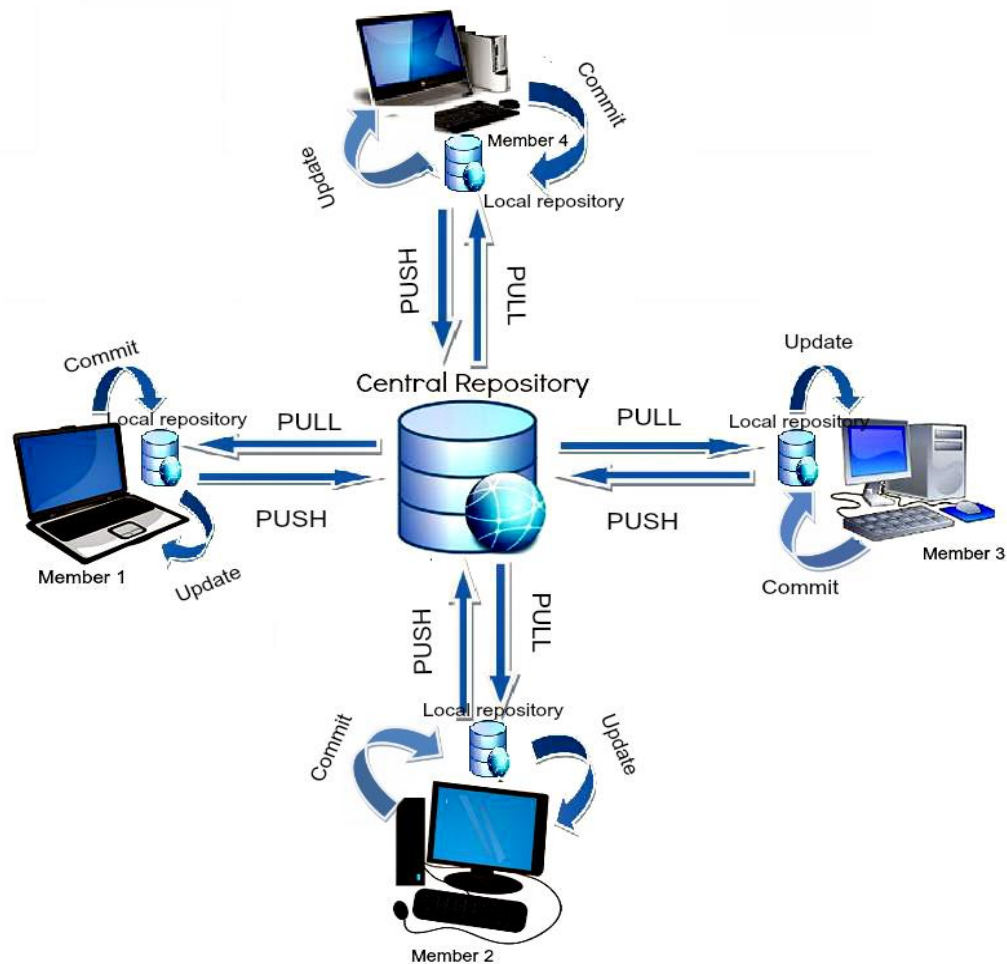
TIPIKUS KÖZPONTI KONFIGURÁCIÓ



ELOSZTOTT MEGVALÓSÍTÁS

- Minden peer „autonóm”
 - Rendelkezik a teljes repository-val (history is)
 - Szinkronizálás nem központosított
 - Peer-ek egymás között cserélhetnek módosításokat (change-set/patch)
 - Nincs egyértelmű, kanonikus, központi kódbázis
 - Legalábbis az eszköz nem kényszeríti ezt ki
 - A gyakori műveletek (commit, history megtekintés, módosítások visszavonása) gyorsak
 - Lokálisan történnek, nincs hálózati kommunikáció egy központi szerverrel
 - Csak akkor szükséges hálózati kommunikáció, ha másik peer-rel szeretnénk szinkronizációt végezni
 - Mivel a teljes repository számos másolata létezik, a stratégia nagyfokú védelmet nyújt az adatvesztés ellen

TIPIKUS ELOSZTOTT KONFIGURÁCIÓ



ÖSSZEHASONLÍTÁS

- Az elosztott megoldásoknak jelenleg nagyobb a lendülete
 - Újabb megközelítés, de ettől még nem feltétlenül jobb
 - Főbb előnyei a központosított megoldásokhoz képest
 - Tipikusan gyorsabbak a verziókövető műveletek
 - Kisebb hálózati forgalom (offline munka)
 - A sok repository másolat biztosítja az adat redundanciát
 - Privát munkavégzés (nem kell mindent nyilvánossá tenni)
 - A centralizált megoldások kissé háttérbe szorulnak, ám ezeknek is van előnyük
 - Az első kódletöltés sokkal gyorsabb, mint elosztott esetben (history, összes módosítás a szerveren marad)
 - Lock mechanizmus (az elosztott rendszerek ezt nem támogatják)
 - Könnyeb megérteni a működését
 - Könnyebb a központi adminisztráció és a felhasználók menedzselése, és kontrolálása

VCS FOGALOMTÁR

- A [terminológia](#) az egyes konkrét rendszerek esetén eltérhet, de a következők általánosságban használatosak
 - Baseline
 - Branch
 - Change
 - Checkout
 - Clone
 - Commit
 - Conflict
 - Delta compression
 - Dynamic stream
 - Export
 - Fetch
 - Forward integration
 - Head
 - Import
 - Initialize
 - Interleaved deltas
 - Label
 - Mainline
 - Merge
 - Promote
 - Pull, push
 - Repository
 - Resolve
 - Reverse integration
 - Revision
 - Share
 - Stream
 - Tag
 - Trunk
 - Update
 - Working copy

KONKRÉT VCS ESZKÖZÖK

- Kliens-szerver
 - Ipari megoldások
 - IBM ClearCase
 - Team Foundation Server
 - Nyílt forrású
 - CVS
 - SVN
- Elosztott
 - Ipari
 - Visual Studio Team Services
 - Plastic SCM
 - Nyílt forrású
 - Git
 - Mercurial
- https://en.wikipedia.org/wiki/List_of_version_control_software



- Concurrent Versions System
- 1990
- Kliens-szerver architektúra
- Egyszerű sorszámmal történő revízió azonosítás
- Verziók összehasonlítása, history, korábbi snapshot lekérése
- Csak merge támogatás (nincs locking)
- Fejlesztési ágak támogatása
- Delta compression használata
- Bináris fájlok minden verzióját tárolja, nincs delta
- Karban tartják, de nincs aktív fejlesztés alatt
 - Nincs komolyabb felhasználó tábor (már)

- Apache Subversion
- CVS utódja („CVS done right”)
- 2000
- Kliens-szerver architektúra
- Commit műveletek atomiak
- Könyvtárak, átnevezések verziózása
- Bináris fájlok támogatása
- Fejlesztési ágak kezelése (branch, tag), olcsó művelet
- File locking + merge
- FSFS adattárolás
 - Az eredeti Berkeley DB leváltása
- Hálózati protokollok: lokális fájl, WebDAV, svn(+ssh)



SVN BRANCH/TAG

- Tipikus SVN mappaszerkezet
 - Trunk
 - Branches
 - Tags
- Trunk
 - Fő ág
- Branch
 - Egyéb fejlesztési ágak
- Tag
 - Tipikusan release
 - Nem módosul (esetleg bugfix)



KI HASZNÁL SVN-T?

- Apache Foundation
- GCC
- GNOME
- KDE
- PuTTY
- Trac
- GnuPG



- Eredeti megalkotója Linus Torvalds
 - Linux kernel verziókövetéséhez
- 2005
- Elosztott architektúra
- Gyors branching/merging támogatása
- HTTP, FTP, SSH támogatás
- Fájl történet kriptográfiai azonosítása
 - Commit azonosító: SHA-1 hash
- Bővíthető merge stratégia
- Mivel elosztott rendszer, szerverként is használható
 - Hozzáférés szabályozására
 - Repository web-es elérésére
 - Repository menedzselésére

KI HASZNÁL GIT-ET?

- Bootstrap
- Node.js
- eclipse
- Ruby on Rails
- eBay
- Twitter
- LinkedIn
- Netflix
- PostgreSQL
- Android

- Eredeti megalkotója Matt Mackall
- 2005
- Elosztott architektúra
- Verziók SHA-1 hash-el történő azonosítása
- HTTP és SSH támogatás
- 3-way merge támogatás
- Python és C
- Unix, Windows, macOS támogatás
- Git-hez nagyon hasonló koncepció



KI HASZNÁL MERCURIAL-T?

- Facebook
- Python
- Mozilla
- OpenSolaris
- Java/OpenJDK
- Bitbucket
- NetBeans
- Nginx
- Octave
- PyPy



- GNU
- 2005
- Elosztott ÉS kliens-szerver architektúra támogatás
 - Központi szerver használata opcionális
 - A két megközelítés egyszerre, egy projekten belül is használható
- Python
- Integrálható más VCS-el
 - Pl. branch készítése SVN-ből
 - Git, Mercurial csak olvasható elérés
 - History export/import: CVS, Darcs, Git, Perforce, Mercurial
- Cross-platform

KI HASZNÁL BAZAAR-T?

- APT
- Emacs
- Mailman
- MxSQL
- Bugzilla
- iPython
- Chromium
- Gawk

VCS ESZKÖZÖK ÖSSZEHAISONLÍTÁSA

- https://en.wikipedia.org/wiki/Comparison_of_version_control_software
- Népszerűség
 - <https://www.openhub.net/repositories/compare>
- Kliens-szerver vs. elosztott / ipari vs. nyílt forrású
- Párhuzamosság feloldási modell (lock vs. merge)
- Támogatott platformok/hálózati protokollok
- Megvalósítás programnyelve
- Tárolási mechanizmus (changeset vs. snapshot)
- Változás egysége (fájl vs. fa)
- Revízió azonosítás (szám, random, hash, stb.)

DIFF ÉS PATCH

- Gyakori fogalmak, egy fájl két verziója közti különbség szöveges leírása
 - Sor alapú „edit distance”
- Hozzáadott és törölt sorok jelölése
- A diff fájlt gyakran patch fájlnek, vagy simán csak patch-nek nevezik
- Alkalmazható egy kódbázisra
 - A művelet után a diff fájlban tárolt módosítások alkalmazásra kerülnek
 - Minden módosult fájl tartalmazni fogja a leírt módosításokat
 - Változások propagálására egyszerűen használható

DIFF ÉS PATCH FORMÁTUM

- Diff előállítás
 - VCS rendszerrel
 - Diff eszközzel
- Fejléc, hozzáadott, törölt sorok jelölése
 - Módosított sorok = törlés + hozzáadás!
- Patch alkalmazása a kiinduló verzió óta történt fájl módosítások esetén
 - Context diff
 - Módosult sorok előtti/utáni kontextus használata
 - Elcsúszott sorszámok kiküszöbölhetők
 - Unified diff
 - Jelenleg leggyakrabban használt formátum
 - Context diff újabb, módosított változata

UNIDIFF PÉLDA

```
1 First line.  
2 Second lin.  
3 Third line.  
4 Fourth line.  
5 5th line.  
6
```

```
1 First line.  
2 Second line.  
3 Third line.  
4 5th line.  
5 Sixth line.  
6
```

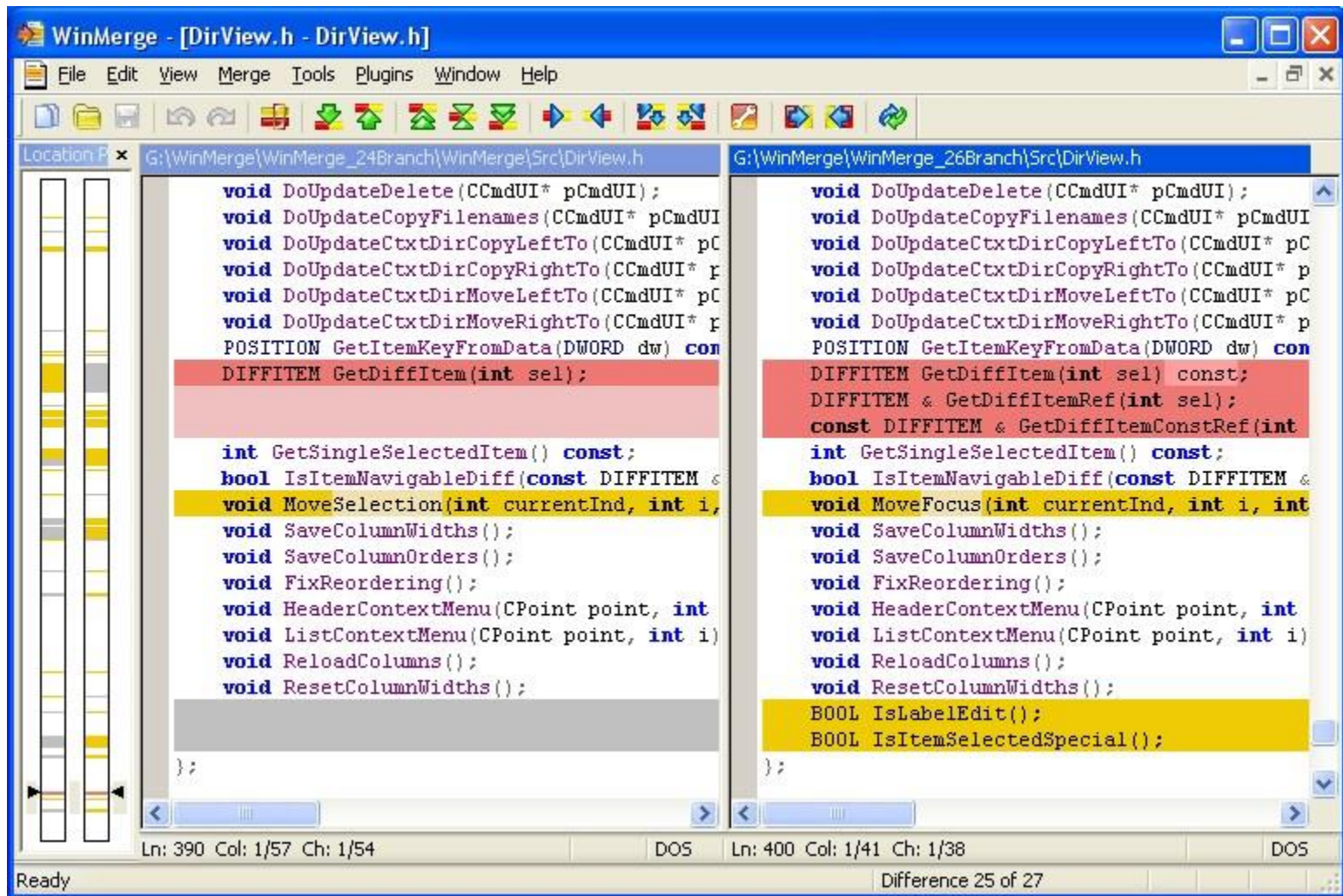
diff

```
diff --git a/old.txt b/old.txt  
index c732c5a..f860556 100644  
--- a/old.txt  
+++ b/old.txt  
@@ -1,5 +1,5 @@  
First line.  
-Second lin.  
+Second line.  
Third line.  
-Fourth line.  
5th line.  
+Sixth line.
```

DIFF ÉS MERGE ESZKÖZÖK

- WinMerge
- Meld
- DiffMerge
- Kdiff
- CodeCompare
- Beyond Compare
- Eclipse compare
- Kompare
- https://en.wikipedia.org/wiki/Comparison_of_file_comparison_tools

WINMERGE PÉLDA



KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

GIT ÉS INTEGRÁLT KOLLABORATÍV PLATFORMOK

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

LEGFONTOSABB GIT PARANCSONK

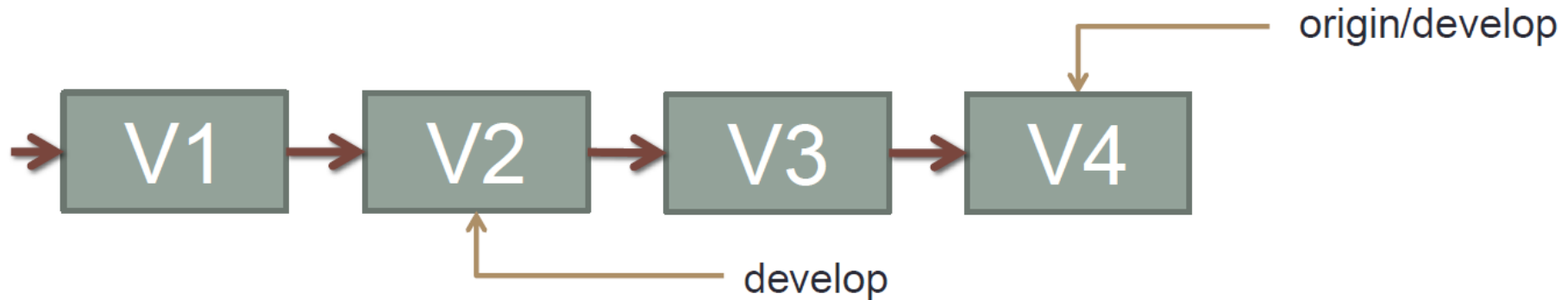
- **git init**
 - Új repó létrehozása
- **git clone**
 - Meglévő repó letöltése
- **git checkout <branch>**
 - Váltás másik fejlesztési ágra
- **git add <files>**
 - Fájlok „stage-elése” commithoz. Létezik *--all* kapcsoló is
- **git commit -m "commit message,"**
 - Új commit
- **git push**
 - Commitok feltöltése remote-ra

LEGFONTOSABB GIT PARANCSOK

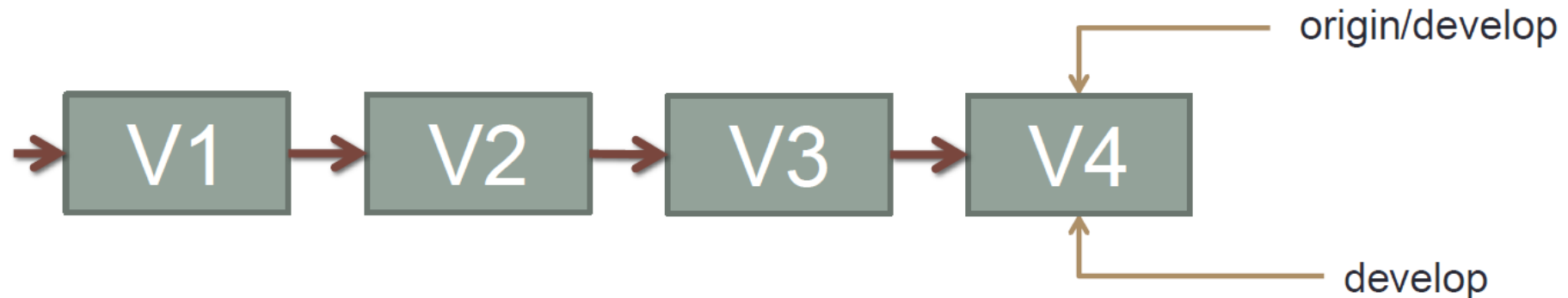
- **git status**
 - Repository információk, például jelenlegi branch, annak állása a remote-hoz képest, módosított fájlok, stb.
- **git log**
 - Commit history megjelenítése
- **git config**
 - Git beállítások módosítása
 - A *--global* kapcsolóval a globális beállítások módosíthatók

USE CASE: LEGÚJABB VÁLTOZAT LETÖLTÉSE

- Probléma: Rég nyúltál ehhez a fejlesztési ághoz, szeretnéd frissíteni a lokális repository-t a legújabb verzióra

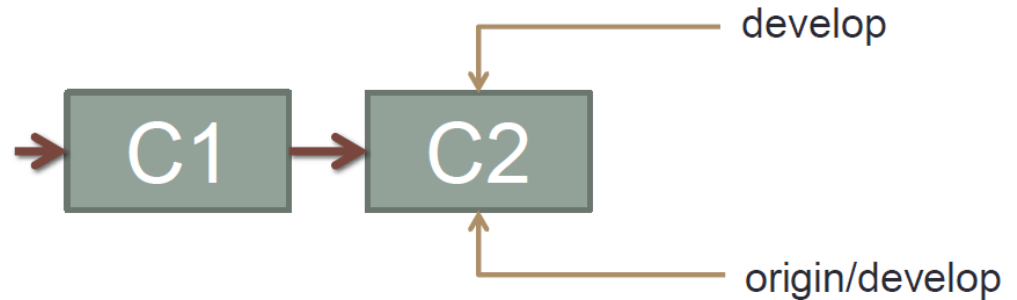


- Megoldás: **git pull**

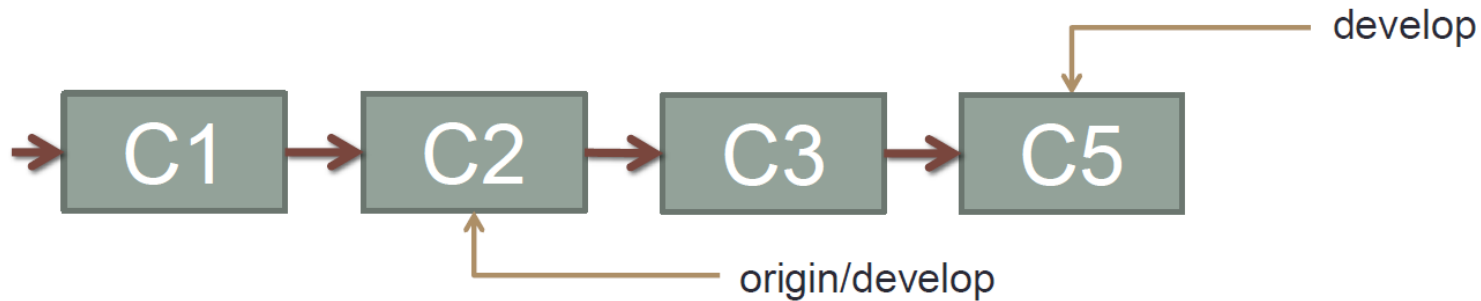


USE CASE: PÁRHUZAMOS FEJLESZTÉS KEZELÉSE

- Kiinduló állapot



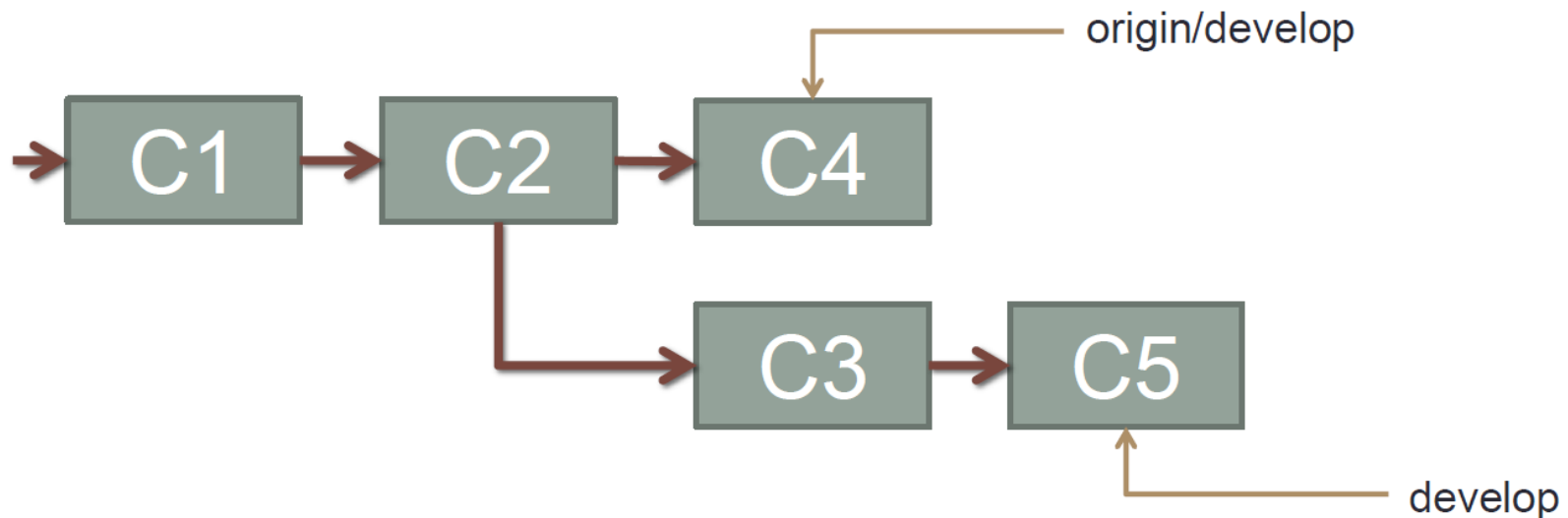
- Valamilyen fejlesztési feladat elvégzése után létrejön két új commit: C3 és C5



- Push** hatására hiba lép fel
 - A *develop* branch eltér az *origin*-től

USE CASE: PÁRHUZAMOS FEJLESZTÉS KEZELÉSE

- Közben valaki más push-olt egy C4-es commitot

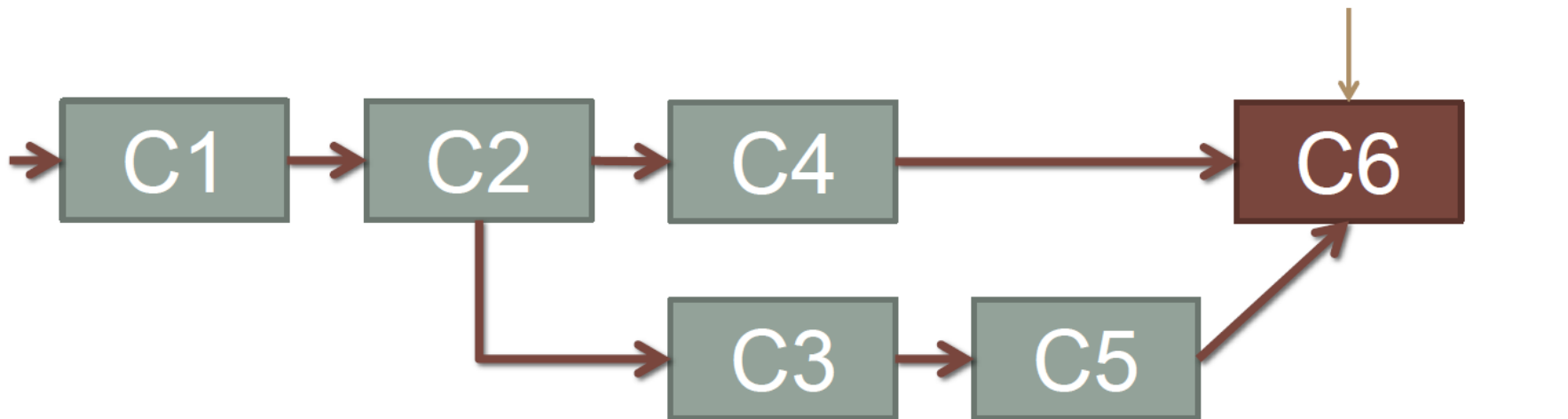


- Reflex
 - *git pull*

USE CASE: PÁRHUZAMOS FEJLESZTÉS KEZELÉSE

- Probléma

- pull = fetch + merge



- Minden esetben jó ez a megoldás?

USE CASE: PÁRHUZAMOS FEJLESZTÉS KEZELÉSE

- A merge commit feleslegesen bonyolítja a commit history-t
- Jó lenne, ha ilyen lenne



- Megoldás: **rebase**
- Workflow
 - git fetch
 - git rebase
- Vagy egy paranccsal
 - git pull --rebase

USE CASE: VÁLTOZÁSOK VISSZAVONÁSA

- Több lehetséges eset
 - **változás nincs commitálva**
 - változás commitálva van, de nincs push-olva
 - változás commitálva és push-olva van
- Három lehetséges módszer
 - **git checkout [-f]**
 - Átmásolja a lokális repóból a fájlokat a working directory-ba
 - **git reset**
 - Törli a bejegyzéseket az indexből (staging area).
 - **git reset --hard**
 - Törli az indexet és visszaállít mindent a legutóbbi verzióra

USE CASE: VÁLTOZÁSOK VISSZAVONÁSA

- Több lehetséges eset
 - változás nincs commitálva
 - **változás commitálva van, de nincs push-olva**
 - változás commitálva és push-olva van
- Három módszer, mindegyik *commit*-ra állít vissza
 - **git reset [--mixed] <commit>**
 - Megmaradnak a változások (a working directory-ban)
 - **git reset --soft <commit>**
 - Megmaradnak a változások (az indexben)
 - **git reset --hard <commit>**
 - Minden törlődni fog ami *commit* után történt

USE CASE: VÁLTOZÁSOK VISSZAVONÁSA

- Több lehetséges eset
 - változás nincs commitálva
 - változás commitálva van, de nincs push-olva
 - **változás commitálva és push-olva van**
- Két lehetőség:
 - History újraírása
 - Új commit a változtatások törlésére

USE CASE: VÁLTOZÁSOK HASONLÍTÁSA

- Mit módosítottunk az utolsó commit óta?
 - **git diff [--staged]**
- Milyen változásokat okozott egy konkrét commit?
 - **git diff <commit>"^"**
 - pl.: **git diff a89b1cc^!** (az " nem minden shellben kötelező)
 - vagy: **git show <commit>** (commit információ megjelenítése)
- Általános alak
 - **git diff <régebbi> <újabb> [--] [path]**
- (de természetesen ennek a parancsnak is számtalan alakja van)

INTEGRÁLT KOLLABORATÍV PLATFORMOK

- Nyilvános host és eszköz platformok
 - Tipikusan nyílt forrású fejlesztéshez
- Integrált környezetet nyújtanak
 - Verzió követés
 - Web alapú elérés is
 - Issue és bug követés (lásd később)
 - Continuous Integration (lásd később)
 - Wiki
 - File hosting
 - Metrikák, analytics
 - Database szolgáltatás

SOURCEFORGE

- <https://sourceforge.net/>
- VCS
 - CVS (törölték)
 - SVN
 - Bazaar (törölték)
 - Git
 - Mercurial
 - Fossil
- Wiki
- MySQL adatbázis
- Projektenként egyedi al-domain
- Levelező lista
- Issue és bug követés, code review

BITBUCKET

- <https://bitbucket.org/>
- VCS
 - Mercurial
 - Git
- Issue követés
 - Jira integráció
- Privát repository-k támogatása
- Python alapú (Django)
- Build rendszer támogatás (lásd később)
 - BitBucket pipelines
- WiKi
- Code review
- REST API

GITLAB

- <https://gitlab.com/>
- VCS
 - Git
- Issue és bug követés
- Code review
- WiKi
- Ruby
 - Később áttértek Go-ra
- Community Edition vs Enterprise Edition
- Beépített CI támogatás
 - GitLab CI
- Privát és publikus felhő alapú felhasználás

LAUNCHPAD

- <https://launchpad.net/>
- VCS
 - Bazaar
 - Git
- Bug követés
- Code review
- Levelező lista
- Specifikáció nyomkövetés
- Q&A és FAQ
- Fordító rendszer
- Ubuntu csomag fordítás és hosting támogatás

GITHUB

- <https://github.com/>
- VCS
 - Git
- Issue és bug követés
- Code review
- Publikus repository-k
- WiKi
- Automatikusan feldolgozott Markdown
- Commit history vizualizáció
- Kiterjedt REST API támogatás
- 3rd party CI támogatás
 - Travis CI

GITHUB REST API

- Szolgáltatások azonosítása URL segítségével:
 - GET, POST, PUT, DELETE
- Lehetséges válasz formátumok
 - XML, JSON, CSV, RSS
- Technológia és platform független
- Lazán csatolt komponensek kommunikációja
- Gyors válasz, sok kommunikációra van szükségünk

GITHUB REST API

- Jelenleg v3 verzió
- Minden API hozzáférés HTTPS-en keresztül megy
- <https://api.github.com>
- Minden adatot JSON formátumba küldünk és fogadunk
- Kliens error
 - Érvénytelen JSON küldése – 400 Bad request
 - Rossz típusú JSON érték küldése – 400 Bad request
 - Érvénytelen mező küldése – 422 Unprocessable Entity

- Három féle hitelesítési módszer használható
 - Basic authentication
 - OAuth2 Token (fejlécben küldés)
 - OAuth2 Token (paraméterként küldés)

PÉLDÁK AZ API HASZNÁLATÁRA

- Felhasználó felfüggesztése

```
const GitHubClient = require('../libs/GitHubClient.js').GitHubClient;  
const users = require('../libs/features/users');
```

```
let githubCli = new GitHubClient({  
  baseUrl:"http://github.at.home/api/v3",  
  token:process.env.TOKEN_GHITHUB_ENTERPRISE  
}, users);
```

```
githubCli.suspendUser({handle:'ripley'})  
  .then(resp => {  
    console.log(resp);  
  })  
  .catch(error => {  
    console.log("error", error)  
  });
```

PÉLDÁK AZ API HASZNÁLATÁRA

- Felhasználó felfüggesztésének törlése

```
const GitHubClient = require('../libs/GitHubClient.js').GitHubClient;  
const users = require('../libs/features/users');
```

```
let githubCli = new GitHubClient({  
  baseUrl:"http://github.at.home/api/v3",  
  token:process.env.TOKEN_GHITHUB_ENTERPRISE  
}, users);
```

```
githubCli.unsuspendUser({handle:'ripley'})  
  .then(resp => {  
    console.log(resp);  
  })  
  .catch(error => {  
    console.log("error", error)  
  });
```

PÉLDÁK AZ API HASZNÁLATÁRA

- Milestone

```
const GitHubClient = require('../libs/GitHubClient.js').GitHubClient;  
const milestones = require('../libs/features/milestones');
```

```
let githubCli = new GitHubClient({  
  baseUrl: "http://github.at.home/api/v3",  
  token: process.env.TOKEN_GHITHUB_ENTERPRISE  
}, milestones);
```

```
githubCli.createMilestone({  
  title: 'Inception',  
  state: 'open',  
  description: 'A discover phase, where an initial problem statement  
and  
functional requirements are created.',  
  due_on: '2016-11-01T09:00:00Z',  
  owner: 'ZeiraCorp', // organization in this case  
  repository: 'toys'  
}).then(milestone => console.log(milestone));
```

GraphQL API V4

- GraphQL – data query language
 - Erősen típusos
 - Introspektív
 - Hierarchikus
 - Specifikus
- A GitHub a v4-es API-nak a GraphQL-t választotta
 - Rugalmasságot biztosít az integrációjuknak
 - Képes arra, hogy pontosan definiáljuk azt az adatot, amit le akarunk kérni
 - Képes több soros REST kérést egyetlen hívással elintézni

HASZNOS LINKEK

- <https://developer.github.com/v3/>
- <https://developer.github.com/v4/>
- <https://github.com/github/platform-samples/tree/master/api/javascript/es2015-nodejs>

KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

BUILD AUTOMATIZÁLÁS ÉS DEPENDENCY MANAGEMENT FOGALMA ÉS ESZKÖZEI

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI 2020 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

BUILD AUTOMATIZÁCIÓ

- Fordítás, építés (build)
 - Folyamat, amely a szoftverfejlesztés során előálló fájlokból és egyéb objektumokból előállítja a végleges, felhasználásra szánt szoftverterméket
 - A folyamat elemei
 - Forráskód fordítása
 - Unit tesztek futtatása
 - Lefordított kód csomagolása (jar, zip, stb.)
 - Telepítők előállítása
 - Adatbázis séma létrehozása, frissítése, adatok aktualizálása
 - Stb., stb., stb.
- Automatizált build esetén a fenti lépések emberi beavatkozás nélkül, megismételhető módon hajtódnak végre a verziókövetőben lévő információk alapján

BUILD AUTOMATIZÁCIÓ

ELŐNYEI

- Folyamatos integráció (Continuous Integration, CI) alapfeltétele (lásd. következő előadás)
- Fordítási folyamat lehetséges hibáinak kiküszöbölése
 - Sok-sok fordítási lépés -> sok-sok hibalehetőség
- A többnyire csak impliciten adott függőségek dokumentálásra kerülnek
 - Pl. a cél környezettel kapcsolatosan
 - Külső könyvtárakkal kapcsolatban
- Fordítási folyamat felgyorsítása
 - Redundáns feladatok eltávolítása
 - Rossz build-ek számának minimalizálása
- A fordítás folyamat, ezáltal a végtermék minőségének javítása

BUILD AUTOMATIZÁCIÓ TÖRTÉNETE

- **1977:** make eszköz Unix rendszerre
 - A szoftver építés folyamatának automatizálása már ekkor sem új ötlet
- **1990-es évek:** a RAD és IDE eszközök számának növekedésével a „make” típusú eszközök száma megnő
 - Vegyes fogadtatásban van részük
- **2000-es évek:** noha az ötlet nem új és nem is specifikus az agilis módszerekre, az agilis megközelítés ezen eszközök újra virágzását hozza el

BUILD AUTOMATIZÁCIÓ

TÍPUSA

- Buil automatizáló eszközök
 - Elsődleges cél a build folyamat automatizálása
 - „Make”-hez hasonló eszközök
 - Konkrétan „make” alapú eszközök
 - Nem „make” alapú eszközök
- Build szkript generáló eszközök
 - A fenti eszközök leíró fájljainak automatikus generálása
- Build automatizáló szerverek (CI)
 - Általános, web-es alapú eszközök
 - A fenti build eszközöket hívják meg valamilyen ütemezésben
 - Fix ütemezés alapján
 - Forráskód változás alapján
 - Cél nem a fordítás automatizálása, hanem a build folyamat rendszeres ellenőrzése (unit test)

KAPCSOLÓDÓ FOGALMAK

- Konfiguráció menedzsment
 - Még magasabb szint
 - Nem csak a build, CI-t foglalja magában
 - Verziókövetés is
 - Személyek
 - Változás követés
 - Stb.
- Csomag kezelő eszközök
 - Telepítő csomagok előállítása/kezelése
 - Függőségek kezelése
 - Központi repository-k használata

BUIL AUTOMATIZÁLÓ ESZKÖZÖK

ÁLTALÁNOS TULAJDONSÁGAI

- A build lépései egy vagy több leíró fájlban vannak megadva
 - Egy központi leíró, több másik leíró
 - A leírók egymásra hivatkozhatnak (pl. include, stb.)
 - A leírók nyelve
 - Egyedi, domain specifikus nyelv (DSL)
 - XML
 - JSON
 - Más szkriptnyelv (pl. Groovy, Python)
- A build leírót egy vagy több binárisból álló program értelmezi és hajtja végre
 - Vagy maga a szkript interpreter

DEPENDENCY MANAGEMENT

- A legtöbb korszerű build automatizáló eszköznek része
- [Dependency hell](#)
 - Software A v1.5 csak lib a >v1.3 és lib b >v2.3.0-al működik
 - lib a >v1.3 számára szüksége lib c 2.x és lib d >1.1.0
 - lib b-nek függősége lib c >2.1
 - lib c függ...
 - lib d függ...
- Dependency management eszköz automatikusan kezeli
 - Explicit könyvtár (3rd party lib) verziók
 - Központi repository-k használata az egyes könyvtárak egyes verzióinak letöltéséhez
 - Függőségek automatikus letöltése
 - Függőségek tranzitív letöltése

DEPENDENCY MANAGEMENT

- Lehet különálló eszköz is
- Tipikusan szkript nyelveknél/browser környezetben, ahol nincs build
 - Interpretáltak, de a függőségek ugyanúgy problémát okoznak
 - JavaScript
 - npm
 - Browserify
 - Jam
 - Bower
 - Python
 - pip
 - PHP
 - Composer
 - .NET
 - NuGet
 - General
 - Nanny

KONKRÉT BUILD ESZKÖZÖK

- Rengeteg eszköz létezik
- Különböző nyelvekhez
- Különböző funkciókkal
- [WiKi lista](#)
 - Make
 - NMake
 - Ant
 - Maven
 - Gradle
 - Buck
 - MSBuild
 - Grunt
 - Stb., stb., stb.

MAKE



- <https://www.gnu.org/software/make/>
- A build automatizáló eszközök „atyja”
- Nyelvfüggetlen
 - Alapvetően C/C++ projektek fordításához használják a gyakorlatban
 - Nem korlátozott egy nyelvre, minden fájl előállításához shell utasítást adunk meg
 - Ami bármilyen fordítót/binárist meg tud hívni
- Makefile tartalmazza a fordítási lépések leírását
 - Minden nem forrásfájl felsorolása
 - Szabályokkal, amelyek leírják, hogyan kell ezeket előállítani más fájlokból
- Fordítást és a program telepítését is el tudja végezni
 - Nem csak csomag építéséhez használható
 - Telepítés, eltávolításhoz, dokumentáció generáláshoz, stb.
- Detektálja mely fájlokat kell újra gyártani a forráskód változás alapján
 - A fájlok újra gyártásának sorrendjét is automatikusan meghatározza
 - Módosítás után nincs teljes újra fordítás

MAKE TARGETS ÉS RULES

- Szabályok – *rules*

- Leírják, hogy hogyan kell bizonyos utasítások sorozatát végrehajtani, hogy egy adott *target* előálljon
- A target *függőségeit* is tartalmazza
- Minden bemeneti fájlt felsorol (forrás vagy más target fájl), amelyek a fenti utasítások bemeneteként szolgálnak

```
target:    dependencies ...  
           commands  
           ...
```

- Make futtatásakor megadhatjuk, hogy melyik target frissüljön
 - Egyébként az első fog
 - Minden olyan target is frissül, amitől a megadott függ
 - Csak az frissül, aminek kell (pl. ha egy target újabb az összes függőségénél, az nem fog)

MAKE PÉLDA

```
edit : main.o kbd.o command.o display.o \  
      insert.o search.o files.o utils.o  
      cc -o edit main.o kbd.o command.o display.o \  
          insert.o search.o files.o utils.o  
  
main.o : main.c defs.h  
      cc -c main.c  
kbd.o : kbd.c defs.h command.h  
      cc -c kbd.c  
command.o : command.c defs.h command.h  
      cc -c command.c  
display.o : display.c defs.h buffer.h  
      cc -c display.c  
insert.o : insert.c defs.h buffer.h  
      cc -c insert.c  
search.o : search.c defs.h buffer.h  
      cc -c search.c  
files.o : files.c defs.h buffer.h command.h  
      cc -c files.c  
utils.o : utils.c defs.h  
      cc -c utils.c  
clean :  
      rm edit main.o kbd.o command.o display.o \  
          insert.o search.o files.o utils.o
```

ANT

- <http://ant.apache.org/>
- Apache Foundation
- Alapvetően Java projektek fordításához
 - De másra is használható
 - Elég általános, akár C/C++ fordítás
 - Akár általános célú szkript
- Maga is Java-ban íródott
 - Jar fájlok + indító szkriptek
- Task alapú leírás egy XML fájlban
 - A végrehajtás „antlib”-ek végzik
 - Rengeteg előre definiált task
 - Saját taskok is készíthetők (saját „antlib”)
- Nem tartalmazza a dependency management-et
 - Apache Ivy-val kombinálható
- Nem feltételez semmilyen projekt könyvtár struktúrát



ANT PÉLDA

```
<project name="MyProject" default="dist" basedir=".>
  <description>
    simple example build file
  </description>
  <!-- set global properties for this build -->
  <property name="src" location="src"/>
  <property name="build" location="build"/>
  <property name="dist" location="dist"/>

  <target name="init">
    <!-- Create the time stamp -->
    <tstamp/>
    <!-- Create the build directory structure used by compile -->
    <mkdir dir="${build}"/>
  </target>

  <target name="compile" depends="init"
    description="compile the source">
    <!-- Compile the java code from ${src} into ${build} -->
    <javac srcdir="${src}" destdir="${build}"/>
  </target>

  <target name="dist" depends="compile"
    description="generate the distribution">
    <!-- Create the distribution directory -->
    <mkdir dir="${dist}/lib"/>

    <!-- Put everything in ${build} into the MyProject-${DSTAMP}.jar file -->
    <jar jarfile="${dist}/lib/MyProject-${DSTAMP}.jar" basedir="${build}"/>
  </target>

  <target name="clean"
    description="clean up">
    <!-- Delete the ${build} and ${dist} directory trees -->
    <delete dir="${build}"/>
    <delete dir="${dist}"/>
  </target>
</project>
```

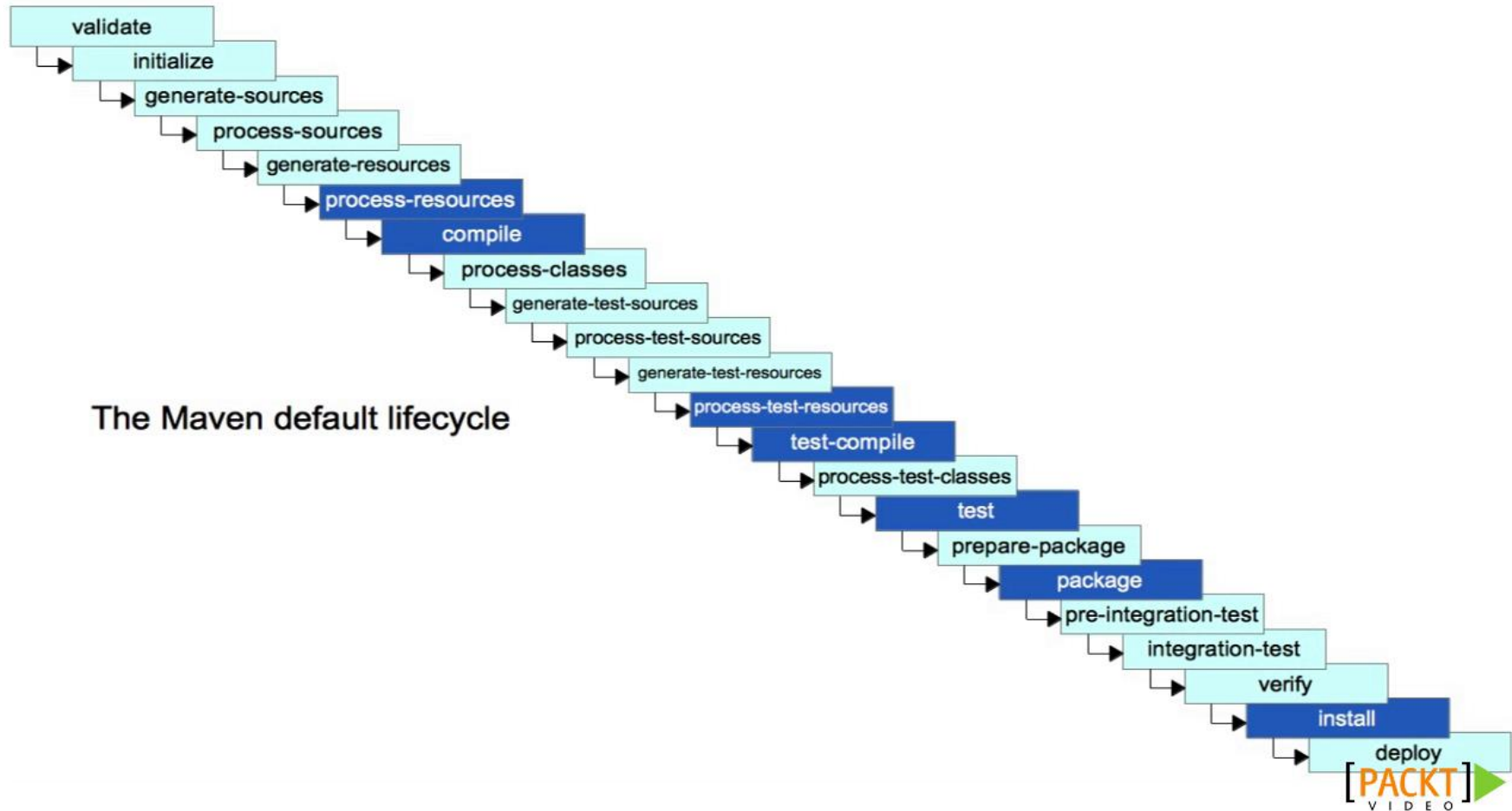


- <http://maven.apache.org/>
- Java build automatizáció
 - Maven, a Yiddish word meaning *accumulator of knowledge*
- Ant koncepcióinak „meghaladása”
 - Pl. függőségek automatikus kezelése
- A fordítás mechanizmusának jó részét elfedi
- Projekt leírása
 - Project Object Model (POM) segítségével
 - XML formátum
 - Plug-in-ek megfelelő halmazával
 - Közös erőforrások minden maven projekt esetén
- Egységes build rendszer
 - Egy projekt maven build-jének megismerése után minden maven build megértése

MAVEN ÉLETCIKLUS ÉS CÉLOK

- Alapkonceptió: lifecycle és goal
- Lifecycle
 - A fordítás egy fix életciklus mentén történik
 - Sok-sok fázisra bomlik (maven definiálja)
- Goal
 - A fázisok végrehajtását úgy módosíthatjuk, hogy az egyes plug-in-ekben definiált célokat (goal) hozzáadjuk
 - Minden fázis meghívja a hozzárendelt goal-okat
 - A goal-ok explicit módon is meghívhatók
 - Parancssorból paraméterezve
- Nem kell a fordítás fázisaival explicit módon foglalkozni

MAVEN ÉLETCIKLUS



MAVEN PROJEKT STRUKTÚRA

```
1. my-app
2. |-- pom.xml
3. `-- src
4.     |-- main
5.         |-- java
6.             |-- com
7.                 |-- mycompany
8.                     |-- app
9.                         |-- App.java
10.    `-- test
11.        |-- java
12.            |-- com
13.                |-- mycompany
14.                    |-- app
15.                        |-- AppTest.java
```

- Forráskód és teszt kód külön mappaszerkezetben
- pom.xml – a projekt objektum modellje
 - Meghatározza, hogy milyen fázisai legyenek a fordításnak
 - Milyen plug-in-eket használunk

MAVEN POM.XML PÉLDA

```
1. <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2.   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
3.   <modelVersion>4.0.0</modelVersion>
4.
5.   <groupId>com.mycompany.app</groupId>
6.   <artifactId>my-app</artifactId>
7.   <version>1.0-SNAPSHOT</version>
8.   <packaging>jar</packaging>
9.
10.  <name>Maven Quick Start Archetype</name>
11.  <url>http://maven.apache.org</url>
12.
13.  <dependencies>
14.    <dependency>
15.      <groupId>junit</groupId>
16.      <artifactId>junit</artifactId>
17.      <version>4.8.2</version>
18.      <scope>test</scope>
19.    </dependency>
20.  </dependencies>
21. </project>
```

MAVEN REPOSITORY

- A POM-xml-ben megadott függőségek egy központi helyről töltődnek le
 - Ez a maven repository
- Külső lib-ek megfelelő verziószámmal, egyedi néven kerülnek fel, és vannak azonosítva
- Fordítás során a függőség bejegyzések alapján a maven ezeket a lib-eket automatikusan azonosítja és letölti
 - Csak első alkalommal
 - Azután az ún. lokális maven repository-ból tölti be
 - „Cache” az eddig használt és letöltött függőségekhez
 - Egy mappa verziószámok és lib nevek alapján strukturálva
 - Csak akkor töltődik le újra fájl, ha a cache-ben nem található meg
- Maven bejegyzések a függőségekhez
 - <https://mvnrepository.com/>

GRADLE



- <https://gradle.org/>
- Nyelvfüggetlen build eszköz
- Groovy-szerű leíró használata
- Task alapú megközelítés
 - Számos beépített task
 - Bővíthető
- A gradle build futtatásához wrapper ajánlott
 - Az aktuális gradle verzió letöltése
 - Windows/Linux futtató szkript generálása

```
task copy(type: Copy) {  
    from 'src'  
    into 'dest'  
}
```

GRADLE TELJES PÉLDA

```
1  apply plugin: 'java'
2
3  repositories {
4      mavenCentral()
5  }
6
7  dependencies {
8      testCompile 'junit:junit:4.12'
9  }
10
11 // START SNIPPET define-main
12 sourceSets {
13     main {
14         java {
15             srcDirs = ['src/java']
16         }
17         resources {
18             srcDirs = ['src/resources']
19         }
20     }
21 // END SNIPPET define-main
22     test {
23         java {
24             srcDir 'test/java'
25         }
26         resources {
27             srcDir 'test/resources'
28         }
29     }
30 // START SNIPPET define-main
31 }
32 // END SNIPPET define-main
```

MIT HASZNÁLNAK A NAGYOK?

- Facebook
 - [Buck](#)
 - Saját fejlesztésű build automatizáló eszköz
 - Nagyfokú párhuzamosítás
- Google
 - [Bazel](#)
 - Eredetileg a Google fejlesztette
 - Egyéb felhasználók
 - Pinetrest
 - Dropbox
 - Huawei
- Microsoft
 - [MSBuild](#)
 - Visual Studio része
- Google Chrome
 - [Ninja](#)
 - Alacsonyszintű leíró, jól párhuzamosítható, más eszközzel kombinált

BUILD LEÍRÓ GENERÁLÓK

- Ahogy láttuk az egyes build lépéseket leíró fájlok/szkriptek összetettek lehetnek
- Bizonyos részei sok esetben ugyanazok
- Variancia az alkalmazott operációs rendszer miatt
 - Kell egy build leíró pl. Windows-ra és Linux-ra
- Megoldás: build leíró generáló eszközök
 - Egyszerű, platform és fordító független konfigurációs fájlok
 - Cél ezekből konkrét build leíró generálása, ami adott build automatizáló eszközzel végrehajtható
 - Konkrét fordítóval
 - Konkrét platformon
 - Konkrét konfigurációban
- [CMake](#), [mmake](#), [automake](#), [cmvn](#), [MPC](#), stb.



- C/C++ rendszerek fordításához generál build leírókat
 - Makefile Unix környezetben
 - MSVC project fájlok Windows esetén (MSBuild-del végrehajtható)
 - Kézzel bővíthető
- CMakeLists.txt
 - Minden könyvtárba
 - Egy vagy több parancsot tartalmazhat
 - COMMAND(args...) formában
 - Ezekből a CMake összeállítja a megfelelő build szkripteket

CMAKE PÉLDA

The first, top-level directory contains the following CMakeLists.txt file.

```
# CMakeLists files in this project can
# refer to the root source directory of the project as ${HELLO_SOURCE_DIR} and
# to the root binary directory of the project as ${HELLO_BINARY_DIR}.
cmake_minimum_required (VERSION 2.8.11)
project (HELLO)

# Recurse into the "Hello" and "Demo" subdirectories. This does not actually
# cause another cmake executable to run. The same process will walk through
# the project's entire directory structure.
add_subdirectory (Hello)
add_subdirectory (Demo)
```

Then for each subdirectory specified, CMakeLists.txt files are created. In the ./Hello directory, the following CMakeLists.txt file is created:

```
# Create a library called "Hello" which includes the source file "hello.cxx".
# The extension is already found. Any number of sources could be listed here.
add_library (Hello hello.cxx)

# Make sure the compiler can find include files for our Hello library
# when other libraries or executables link to Hello
target_include_directories (Hello PUBLIC ${CMAKE_CURRENT_SOURCE_DIR})
```

Finally, in the ./Demo directory, the third and final CMakeLists.txt file is created:

```
# Add executable called "helloDemo" that is built from the source files
# "demo.cxx" and "demo_b.cxx". The extensions are automatically found.
add_executable (helloDemo demo.cxx demo_b.cxx)

# Link the executable to the Hello library. Since the Hello library has
# public include directories we will use those link directories when building
# helloDemo
target_link_libraries (helloDemo LINK_PUBLIC Hello)
```

- Maven POM leírók generálása
- Egyszerű, deklaratív leírók alapján
- JVM-re fordításnál tipikusan nincs „configure” lépés a build előtt
 - Cmvn ezt nyújtja
- Build konfiguráció leírás alapján konkrét build szkriptek generálása
 - Konfiguráció változás esetén újragenerálás
 - Fordítást is elvégzi
 - Jelenleg Maven támogatott
 - Tervben van Ant, Ivy, SBT, Sbuild támogatás
- Fenti Maven POM generálásához szükséges Cmvn konfiguráció

```
project: com.mycompany.app:my-app:1.0-SNAPSHOT;packaging=jar
test: junit:junit:4.8.2
```

BUILD ESZKÖZÖK ÖSSZEHASONLÍTÁSA

- Az eddig tárgyalt eszközök leíró nyelvének és licenszeinek összehasonlítása
- https://en.wikipedia.org/wiki/List_of_build_automation_software
- Leírók
 - XML
 - Ruby
 - Clojure
 - Groovy
 - Scala
 - Python
 - JSON
 - Stb.

KONKRÉT DEPENDENCY MANAGEMENT ESZKÖZÖK

- A legtöbb modern build eszközbe integrálva van
- De nem mindegyikbe, ezekhez külön függőség kezelő eszközt használhatunk
 - Pl. Ant + Ivy
- Szkript nyelvek esetén nincs (korlátozottan van) értelme a build-nek
 - Függőségek viszont ott is vannak, amik futáskor derülnek ki
- Webes környezetben szintén nincs build
 - De függőségek igen

APACHE IVY

- <http://ant.apache.org/ivy/index.html>
- Java orientált dependency manager eszköz
- Ant-tal szorosan integrált
 - Az Ant struktúráját követi
 - Az Ant projekt alprojektje
- Ant + Ivy -> Maven jellegű funkcionalitás Ant-hoz
- Függőségi riportok
- Függőségek automatikus letöltése a lib könyvtárba
 - Tranzitív függőségek feloldása
 - Futtatáshoz nem kell az Ivy
- Bővíthető
- Maven repository támogatás



APACHE IVY PÉLDA

- Ivy leíró XML

```
<ivy-module version="2.0">
  <info organisation="org.apache" module="hello-ivy"/>
  <dependencies>
    <dependency org="commons-lang" name="commons-lang" rev="2.0"/>
    <dependency org="commons-cli" name="commons-cli" rev="1.0"/>
  </dependencies>
</ivy-module>
```

- Maven megfelelője

```
<dependency>
  <groupId>commons-lang</groupId>
  <artifactId>commons-lang</artifactId>
  <version>2.0</version>
</dependency>
```

- Használat Ant build fájlban

```
<project xmlns:ivy="antlib:org.apache.ivy.ant" name="hello-ivy" default="run">
  ...
  <!-- =====
    target: resolve
    ===== -->
  <target name="resolve" description="--> retrieve dependencies with ivy">
    <ivy:retrieve />
  </target>
</project>
```



- <https://www.npmjs.com/>
- JavaScript csomagkezelő
- JS csomagok/modulok kezelése
 - Letöltése
 - Frissítése
- JS csomag
 - Egy JS mappa, benne egy `package.json` fájlal
- JS modul
 - Bármilyen, amit a `require()` segítségével be lehet tölteni egy Node.js programban

```
var req = require('request')
```



- <https://github.com/pypa/pip>
- Python Packaging Authority által javasolt m Python függőségek telepítéséhez
- Függőségek telepítése több forrásból
 - [PyPi](#) (Python Package Index)
 - VCS
 - Lokális projekt
 - Disztribúció fájlok
- Közvetlen modul név/verzió alapján történő telepítés
- Requirements file alapján
 - Több modul listája, amelyek pip-el telepíthetők
 - Verziószám megkötések

PIP REQUIREMENTS FILE PÉLDA

Use `pip install -r example-requirements.txt` to install:

```
#
##### example-requirements.txt #####
#
##### Requirements without Version Specifiers #####
nose
nose-cov
beautifulsoup4
#
##### Requirements with Version Specifiers #####
# See https://www.python.org/dev/peps/pep-0440/#version-specifiers
docopt == 0.6.1           # Version Matching. Must be version 0.6.1
keyring >= 4.1.1          # Minimum version 4.1.1
coverage != 3.5           # Version Exclusion. Anything except version 3.5
Mopidy-Dirble ~ 1.1       # Compatible release. Same as >= 1.1, == 1.*
#
##### Refer to other requirements files #####
-r other-requirements.txt
#
#
##### A particular file #####
./downloads/numpy-1.9.2-cp34-none-win32.whl
http://wxpython.org/Phoenix/snapshot-builds/wxPython_Phoenix-3.0.3.dev1820+49a8884-cp34-none-win_
#
##### Additional Requirements without Version Specifiers #####
# Same as 1st section, just here to show that you can put things in any order.
rejected
green
#
```

RUBYGEMS



- <https://rubygems.org/>
- Ruby csomagkezelő
- „gem”
 - Tetszőleges kód, ami a Ruby környezetben fut
 - Parancssori program
- Ruby kódban elhelyezett `require` utasítás
 - A hivatkozott gem letöltése, és megfelelő helyre történő elhelyezése

```
% irb -rpp
>> require 'ap'
=> true
>> pp $LOAD_PATH.first
".../gems/awesome_print-1.0.2/lib"
```

KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

FOLYAMATOS INTEGRÁCIÓ (CONTINUOUS INTEGRATION) ÉS CODE REVIEW FOGALMA ÉS ESZKÖZEI

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap

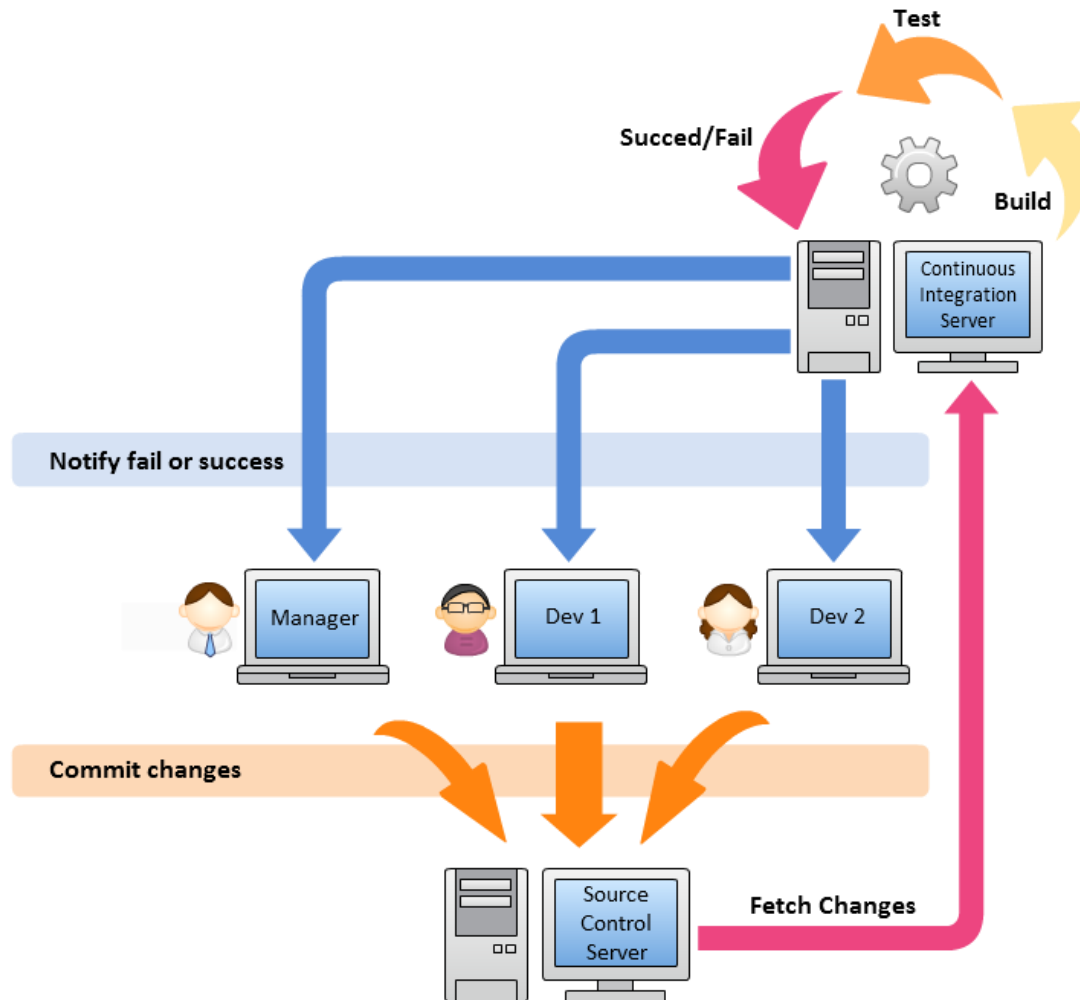


BEFEKTETÉS A JÖVŐBE

FOLYAMATOS INTEGRÁCIÓ

- Összes fejlesztői forráskód változat folyamatos összefésülése
 - Egy központi, közös verzióba
 - Napi többszöri merge
 - Az összefésült változat build-je
- 1991-ben jelent meg a kifejezés
 - Grady Booch
 - Napi szintű integrációt javasoltak
- Később az XP model adoptálta
 - Bevezette a napi többszöri integrációt
 - Akár napi néhány 10-szer is

FOLYAMATOS INTEGRÁCIÓ



FOLYAMATOS INTEGRÁCIÓ

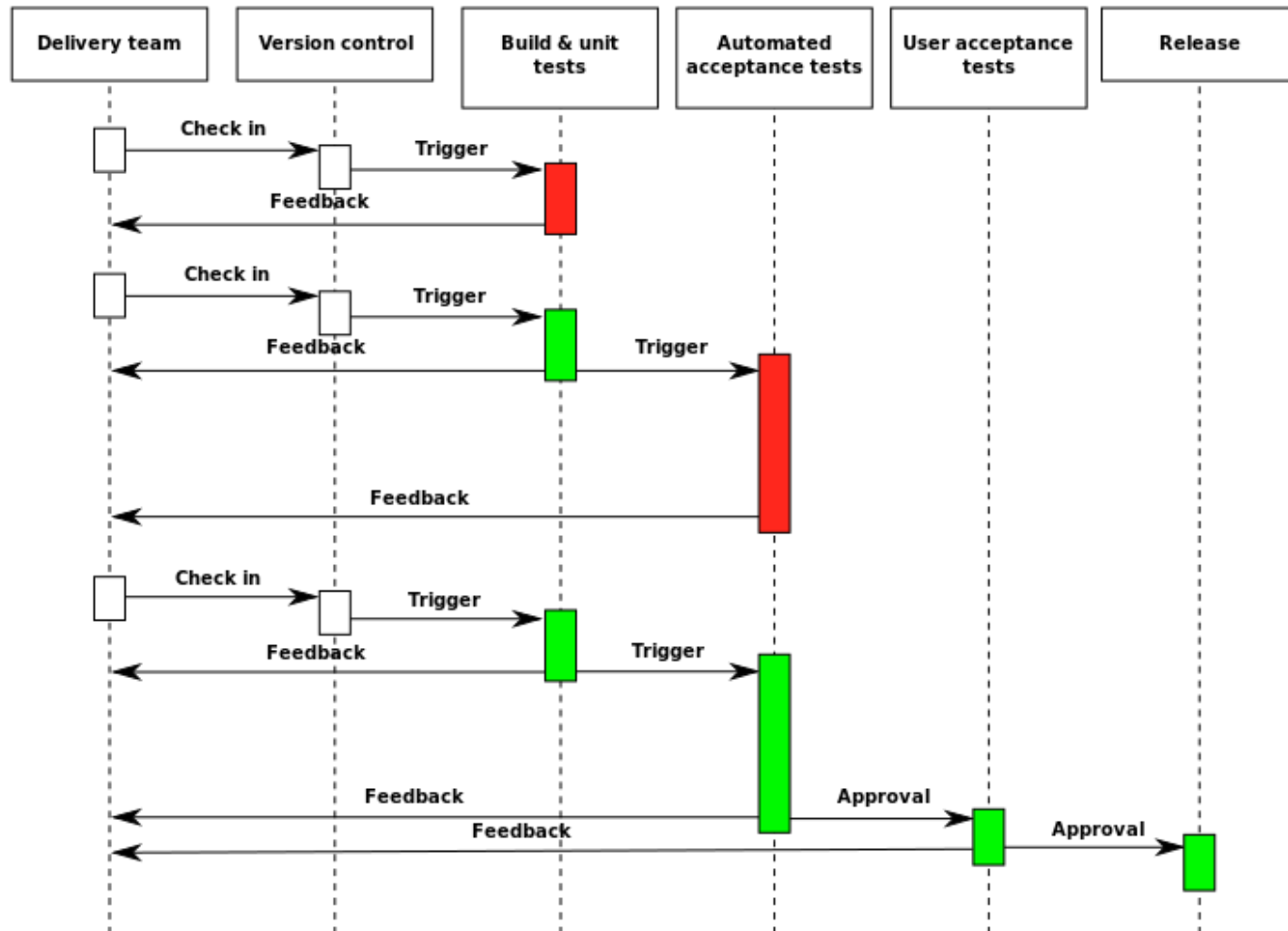
CÉLJA

- Integrációs problémák kiküszöbölése
 - „Integration hell”
- Központi szerver, mely biztosítja
 - A forráskódból mindig előállíthatók a szükséges állományok
 - Tesztek futtatása, amely segít az alap funkciók működési problémáit feltárni
 - Unit tesztek
 - Integrációs tesztek
 - Egyéb periodikusan elvégzendő feladatok
 - Minőségbiztosítás (statikus/dinamikus elemzés)
 - Teljesítmény mérés/profilozás
 - Dokumentáció generálása forráskód alapján
- Csökkenti a termék piacra kerülési idejét
 - Continuous deployment (CD) tovább megy, minden kód módosítás után telepíthető a rendszer ügyfelek számára

FOLYAMATOS INTEGRÁCIÓ A GYAKORLATBAN

- VCS-el összehangolt működés
 - Nem periodikus integráció, hanem minden egyes commit után
 - A trigger automatikus, a CI szerver figyeli a VCS-t (vagy a VCS indítja a CI build-et)
 - Fejlesztők gyakran commit-álnak
- A rendszer építése automatizált
 - Lásd build automatizáló eszközök
 - Tesztelés integrálva, minden sikeres build után lefutnak
 - A build folyamatnak gyorsnak kell lennie
- A tesztelés az éles környezet replikáján történik
- Nagyon gyorsan előállíthatók az átadandók (binárisok, telepítők, dokumentáció, stb.)
- Nagyon könnyen kideríthető, ha a rendszer nem fordul
- Nem csak integráció, hanem automatikus deployment is történik a fordítás után (CD)

FOLYAMATOS INTEGRÁCIÓ A GYAKORLATBAN



FOLYAMATOS INTEGRÁCIÓ ELŐNYEI

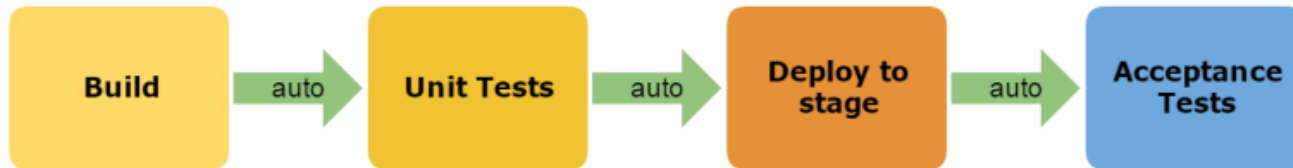
- Integrációs hibák korai felderítése
- Kiadás előtti káosz elkerülése
- Aktuális build állandóan elérhető
 - Teszteléshez
 - Demohoz
 - Kiadáshoz
- Gyakori check-in rákényszeríti a fejlesztőket a moduláris, kevésbé komplex kód előállítására
- A tesztelés futtatása miatt
 - Azonnali visszajelzés egy módosítás negatív hatásáról
 - Tesztelési és QA mérőszámok elérhetősége
 - Kód lefedettség
 - Forráskód metrikák

FOLYAMATOS INTEGRÁCIÓ HÁTRÁNYAI

- Automatizált tesztkörnyezet előállítása időigényes és költséges
- Automatizált build környezet felállítása erőforrásigényes lehet
 - De ezt egyébként is elvégeznénk
- Nem mindig éri meg használni
 - Pl. nagyon kicsi projekt
 - Nem tesztelhető legacy kód
- A hozzáadott értéke nagyban függ a kapcsolódó tesztek minőségétől
- Nagyon nagy csapat esetén a gyakori chek-in miatt a CI szerver leterhelt lehet, ami a folyamatokat lassíthatja
- Részleges feature feltöltése hibát okozhat

CONTINUOUS INTEGRATION/DELIVERY/DEPLOYMENT

Continuous Integration



Continuous Delivery



Continuous Deployment



FOLYAMATOS INTEGRÁCIÓ ESZKÖZEI

- Szerver komponensek
 - CS/build server
 - Fizikai vagy virtuális gép
 - VCS kapcsolat
 - Build automatizáció
- Egyéb funkciók
 - Kód analízis
 - Kód lefedettség
 - Kódminőség riportok
 - Pipeline-ok
 - Build összehasonlítás
 - IDE integráció
 - Egyéb eszközök támogatása

FOLYAMATOS INTEGRÁCIÓ ESZKÖZEI

- **Nyílt forrású/ingyenes eszközök**
 - Jenkins
 - Go CD (szakmai segítségnyújtás fizetős)
- **Ipar/fizetős eszközök**
 - TeamCity (van ingyenes, korlátozott változata)
 - Travis CI (GitHub open source projektekhez és az első 100 build-hez ingyenes)
 - Bamboo
 - GitLab CI (van ingyenes, próbaverziója)
 - CircleCI (van ingyenes, próbaverziója)
 - Codeship (havi 100 build ingyenes)
 - Codefresh (havi 200build, 5 párhuzamos build, 1 környezet ingyenes)

JENKINS

- <https://jenkins.io/>
- Nyílt forrású, 2011-ben jelent meg
- Java nyelven írt
- A Hudson CI eszközből származik
- Alapból Apache Tomcat konténerben fut
- VCS támogatás
 - AccuRev
 - CVS
 - SVN
 - Git
 - Mercurial
 - Perforce
 - ClearCase
- Build támogatás
 - Ant
 - Maven
 - Shell/batch szkript
 - + számos plug-in



TRAVIS CI

- <https://travis-ci.org/>
- Hosted megoldás
 - CI szolgáltatás GitHub projektekhez
- Saját telepítés fizetős
- Ruby nyelven írt
- YAML alapú konfiguráció
- Minden GitHub commit után lefut
 - Branchek is konfigurálhatók
 - Értesítés e-mailben vagy IRC üzenetben
- Külső eszközökkel integrálható
 - Lefedettség mérő (coverage), statikus elemző
- Számos nyelv fordításának támogatása



Travis CI



- <https://www.jetbrains.com/teamcity/>
- JetBrains fejleszti (IntelliJ, PyCharm, stb.)
- Java nyelven írt, 2006
- Fizetős
- Extra szolgáltatások
 - Gated commit (commit előtti build futtatás)
 - Build grid
 - Integrált kódfedettség, inspekció és duplikátum keresés
 - IDE integráció
 - Java, .NET, Ruby támogatás
- VCS támogatás
 - SVN, Perforce, CVS, Git, Mercurial, Visual Studio Team Services, stb.

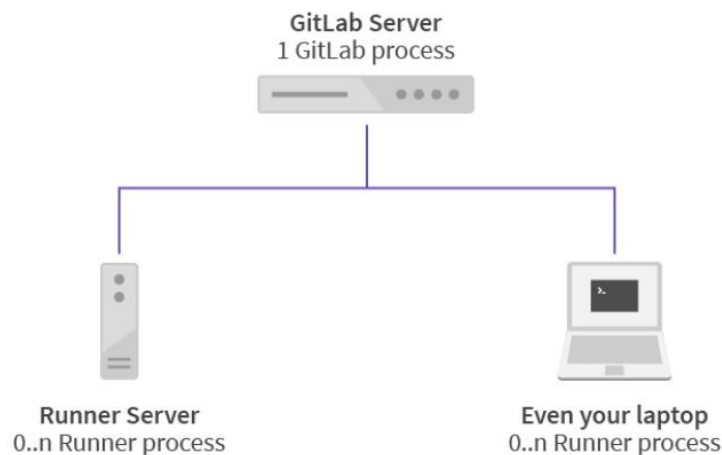
GITLAB CI

- <https://about.gitlab.com/features/gitlab-ci-cd/>
- GitLab szerves része
- Nyílt forráskódú
- Skálázható
- Multi-platform
 - Java, PHP, Ruby, C, stb.
- Párhuzamos build
- Pipeline mechanizmus
- Docker támogatás
- GitLab Runner
 - Build futtatását végzi
 - Go-ban íródott



GitLab CI

Open Source Continuous Integration made easy



CI ESZKÖZÖK ÖSSZEHASONLÍTÁSA

- https://en.wikipedia.org/wiki/Comparison_of_continuous_integration_software
- Platform
- Licenz
- Támogatott build eszközök
- Értesítés küldés
- IDE integráció
- Egyéb integráció
- VCS támogatás

MIT HASZNÁLNAK A NAGYOK?

- Nincs sok nyilvános adat
- Sokan egyedi, saját megoldásokat használnak
- A következők azonban gyakoriak
 - Jenkins
 - TeamCity
 - Bamboo
 - Buildbot
 - Go

www.infoq.com/research/ci-server

Average Analysis

Total number of participants: 789

Option	Adoption	Relevance	Votes	Details
Jenkins	85%	89%	569	Heatmap 
TeamCity	72%	81%	192	Heatmap 
Bamboo	68%	73%	164	Heatmap 
Hudson	62%	69%	130	Heatmap 
Cruise Control	56%	54%	129	Heatmap 
Go	63%	75%	117	Heatmap 
Team Foundation Server	67%	65%	108	Heatmap 
TravisCI	64%	75%	98	Heatmap 
Anthill	62%	39%	43	Heatmap 
Buildbot	56%	61%	37	Heatmap 
Continuum	60%	57%	32	Heatmap 
QuickBuild	61%	62%	18	Heatmap 
FinalBuilder Server/Continua CI	66%	62%	17	Heatmap 
Mojo	59%	47%	11	Heatmap 
Parabuild	45%	39%	11	Heatmap 
Gump	56%	38%	11	Heatmap 
Zed	46%	44%	11	Heatmap 
easyCIS	46%	40%	10	Heatmap 
Sin	47%	48%	9	Heatmap 
Pulse	35%	50%	9	Heatmap 

KÓD ÁTVIZSGÁLÁS (CODE REVIEW)

- A minősegbiztosítás másik zászlóshajója
 - CI és tesztelés mellett/előtt
- Forráskód változtatások kézi ellenőrzése
 - Fejlesztési folyamat részeként
 - Mielőtt új kód bekerülhetne a VCS/CI vérkeringésbe
 - Fejlesztők egymás kódját ellenőrzik
 - Minőségjavítás
 - Hibák korai feltárása
- Code review típusai
 - Walkthrough
 - Technical review
 - Inspection

WALKTHROUGH

- Nem formális folyamat
- A forráskód szerzője vezeti
 - Elmagyarázza az átvizsgálandó kód/dokumentum tartalmát
- Célja inkább a tudás transzfer
- A résztvevők beavatása
- A megoldás átbeszélése, helyességének validálása

TECHNICAL REVIEW

- Kevésbé formális folyamat
 - Pontosabban bármi lehet a teljesen informálistól a formálisig
- Egy kiképzett moderátor vezeti
- Gyakran peer review, a menedzsment részvétele nélkül
- Célok a hibák korai feltárása
- Korai technikai irányvonalak tisztázása
- A résztvevők technikai informálása

INSPECTION

- Leginkább formális folyamat
- Egy gyakorlott moderátor vezeti
- A review-ra szánt dokumentumok/kód már a review előtt átnézésre kerülnek
 - A közös megbeszélésen ezeket egyeztetik
- A folyamat során talált hibák naplózásra, lejegyzésre kerülnek
- A review után egy formális utókövetés (follow-up) is megtörténik a moderátor által
- A cél általában a kód/dokumentum minőségének javítása
- Hibák felfedezése
- Információ csere

CODE REVIEW ESZKÖZEI

- Tipikusan web alapú kollaboratív eszközök
 - „Offline” kód átvizsgálás
- Fejlesztők egymás forráskód módosításait áttekinthetik
 - Egy diff-hez hasonló webes nézeten
 - Megjegyzések beszúrása, akár kódsor szinten
- Tipikus eszközök
 - Gerrit
 - Rietveld
 - GitLab
 - Codacy
 - Upsource

GERRIT

- <https://www.gerritcodereview.com/>
- Rietveld fork-ja
- Code review eszköz Git-hez
- Web alapú megjelenítés
- Régi és új forráskód megjelenítése diff nézetben
- Sor alapú kommentelés
- Git repository-k kezelése
- Plug-in támogatás
- Java + GWT (JS) megjelenítés



GERRIT KÓD NÉZET

gerrit / gerrit-server/src/main/java/com/google/gerrit/server/change/PatchSetInserter.java

```
106 private PatchSet patchSet;
107 private ChangeMessage changeMessage;
108 private SshInfo sshInfo;
109 private ValidatePolicy validatePolicy = ValidatePolicy.GERRIT;
110 private boolean draft;
111 private boolean runHooks;

112 private boolean sendMail;
113 private Account.Id uploader;
114 private BatchRefUpdate batchRefUpdate;
115
116 @Inject
117 public PatchSetInserter(ChangeHooks hooks,
118     ReviewDb db,
```

```
110 private PatchSet patchSet;
111 private ChangeMessage changeMessage;
112 private SshInfo sshInfo;
113 private ValidatePolicy validatePolicy = ValidatePolicy.GERRIT;
114 private boolean draft;
115 private boolean runHooks = true;

116 private boolean sendMail = true;
117 private Account.Id uploader;
118 private BatchRefUpdate batchRefUpdate;
119
120 @AssistedInject
121 public PatchSetInserter(ChangeHooks hooks,
122     ReviewDb db,
```

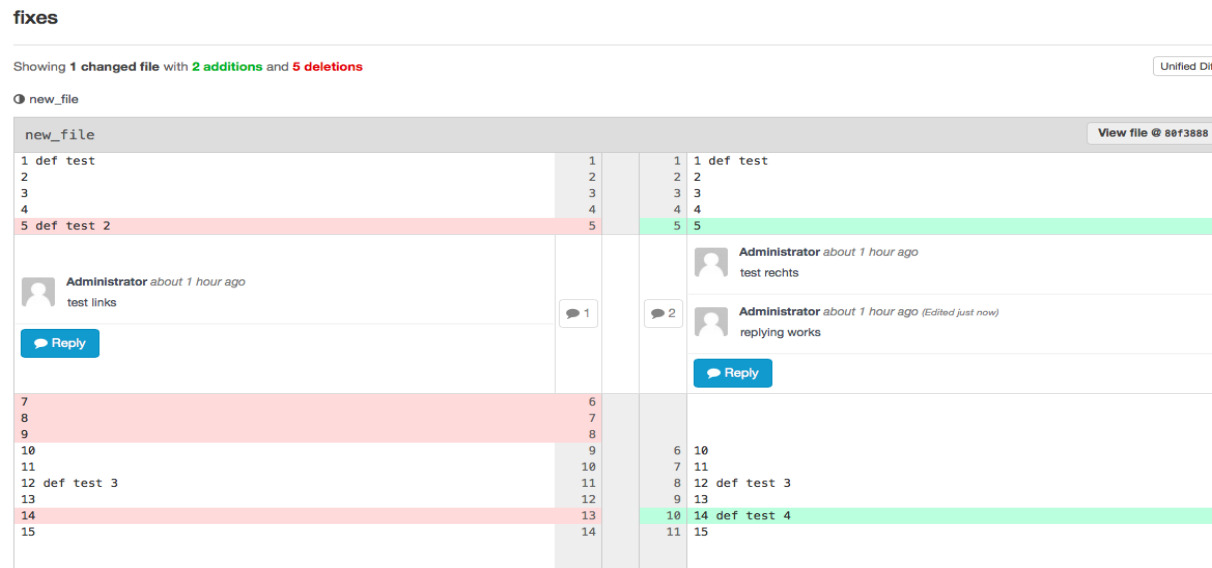
Stefan Beller Why do you move this out of the constructor? Initially I assumed this... Jan 28 2:55 PM
Dave Borowitz Because it would be identical between the two constructors, so it sa... Jan 28 3:19 PM

GERRIT FELHASZNÁLÓI

- Android
- Chrome OS
- eBay
- Eclipse Foundation
- Garmin
- Go
- GWT
- LibreOffice
- OpenStack
- Qt



- <https://about.gitlab.com/features/#review>
- GitLab része
- Jóváhagyás mechanizmus
- Inline kommentezés





- <https://www.codacy.com/>
- Szokásos funkcionalitás +
 - Statikus elemzés
 - Python, Ruby, PHP, Java, JS, Scala, Swift, TypeScript támogatás
 - Git integráció
 - Kód minőség osztályozás minden commit-ra
- Magas konfigurálhatóság
- Nyílt, ingyenes
- Más eszközökkel való integrálhatóság
 - Jenkins, GitHub, Bitbucket, JIRA, stb.

CODACY FELÜLET

```
48 +  
49 +   for {  
50 +       comparisonValues <- comparisonValues  
51 +   } yield {  
52 +       val (good, bad) = comparisonValues.partition{ case FileValue(filename, value) => value >= 0 }
```



cafreeman



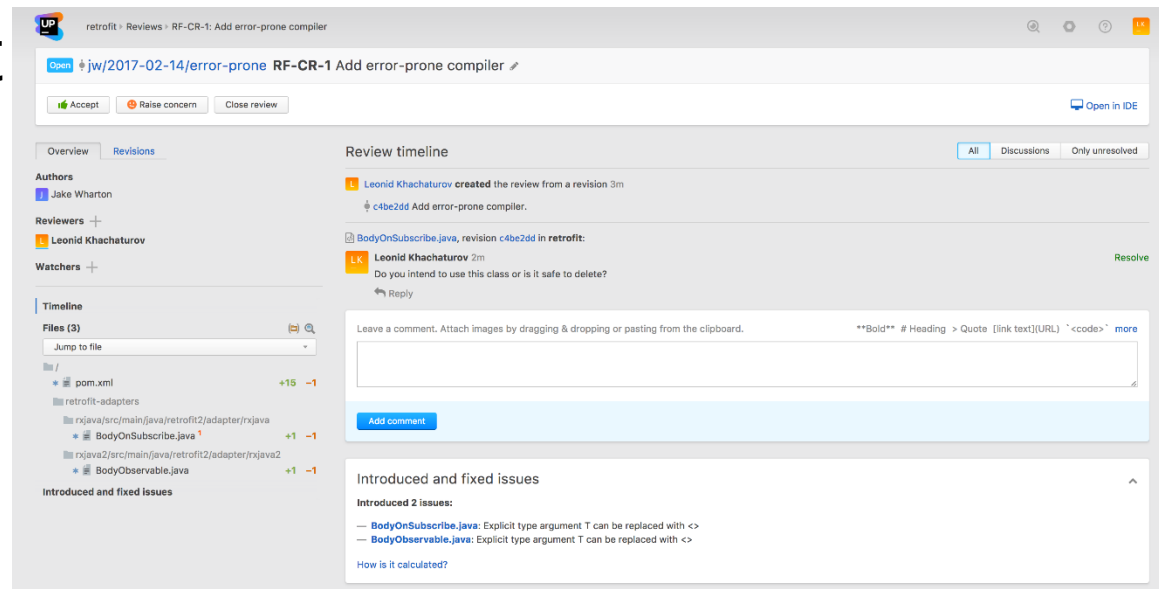
Issue found: [Default case not specified.](#)

Reply • Delete • Create task • 5 days ago

```
53 +   GenericDeltaInfo(good, bad)  
54 + }  
55 + }  
56 +
```



- <https://www.jetbrains.com/upsource/>
- Code review és repository böngésző
- IDE integráció
 - Kód módosítások megtekintése az IDE-ben
- Syntax highlight
- Pull request
- Branch review



EGY VALÓS CODE REVIEW + CI FOLYAMAT



openstack®

OpenStack felhő platform

Nyílt forrású, teljesen nyitott közösség és fejlesztés

Több millió Python kódsor

Több ezres fejlesztői bázis

Infrastruktúra

Git

Jenkins

Gerrit

Launchpad

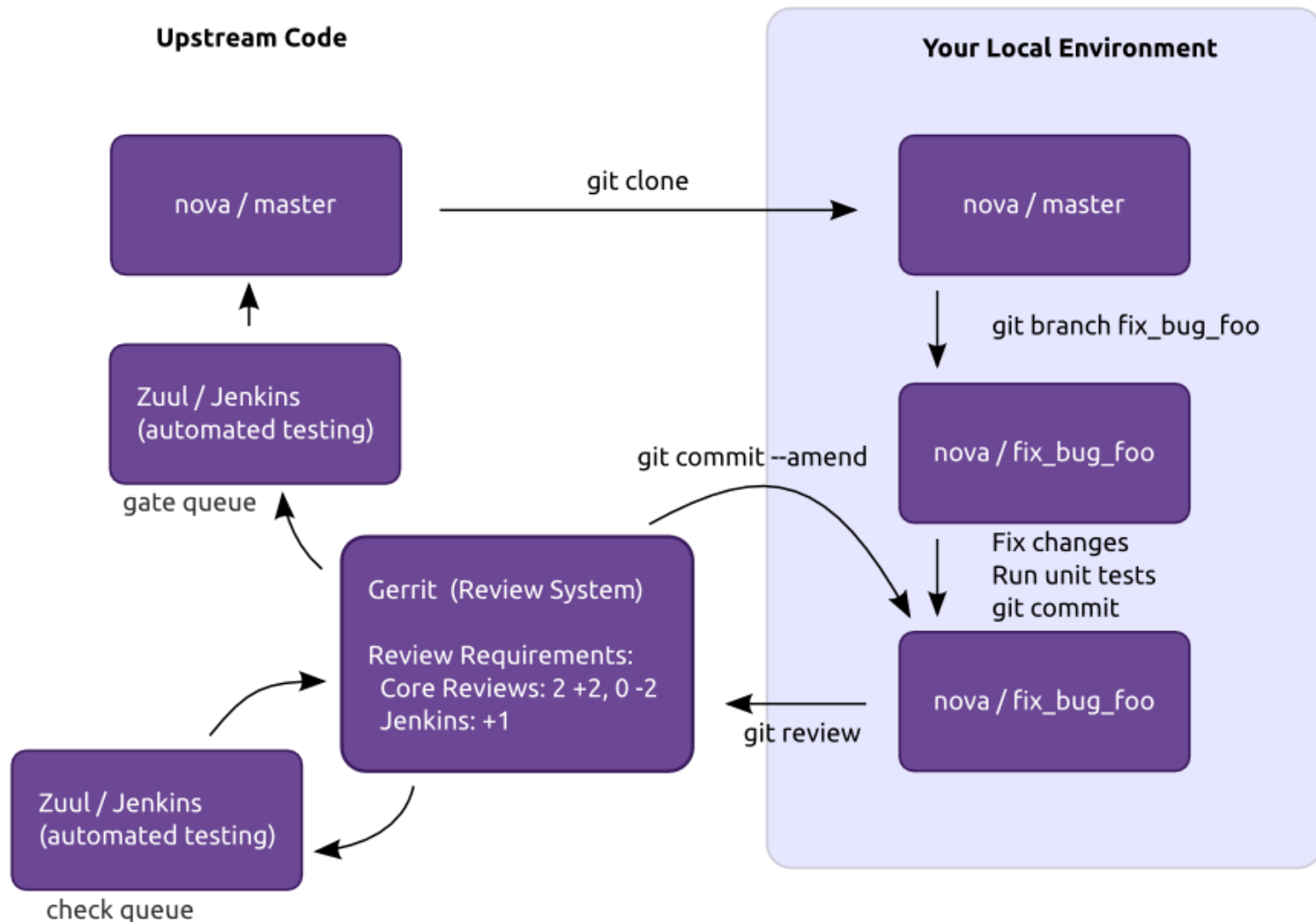
Kód módosítása patch-ek formájában

Csak jóváhagyott kód merge-elődik a fő ágba

OS CODE REVIEW + CI FOLYAMAT

- Sikeres kontribúció lépései
 - Kód letöltése
 - Kód módosítása
 - Lokális gépen tesztek futtatása (tox)
 - Módosítási javaslat felterjesztése (gerrit)
 - Automatikus Jenkins build (+1)
 - Unit tesztek
 - Funkcionális tesztek
 - Code review más fejlesztőktől (+1)
 - Code review core tagoktól (+2)
 - 2 darab után Workflow+1
 - Újabb Jenkins elemzés (gate job) (+2)
 - Integrációs és platform tesztek
 - Sikeres Jenkins build után a kód merge-elése

OS CODE REVIEW + CI FOLYAMAT



KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

PROJEKTMENEDZSMENT ÉS ESZKÖZEI – ISSUE ÉS BUG KÖVETÉS

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

ISSUE KÖVETÉS

- Fejlesztés és menedzsment fontos tevékenysége
- Szoftveres támogatás (ITS – issue tracking system)
 - Issue – bármilyen ügyféltől, vagy belső alkalmazottól érkező bejelentés
 - Lehet hiba riport, feladat, kérés, stb.
 - Szisztematikusan összegyűjti és rendszerezi egy szoftvertermékhez kapcsolódó információkat
- Kiegészítő információk az egyes issue-khoz
 - Prioritás
 - Érintett komponens
 - Tervezett kezdési/befejezési dátum
 - Felelős személyek
 - Részletes leírás
 - Stb.

ITS-EK FŐ FUNKCIÓI

- Hibák, kérések rögzítése
- A rögzített issue-k hozzárendelése a megfelelő személyekhez
- Az egyes issue-kra fordított idő, munkaminőség monitorozása
- Hibaszámok elemzése, jelentések készítése
- Issue bejelentések automatikus generálása
- Kérdések/válaszok (FAQ) szisztematikus összegyűjtése
- Issue-k prioritizálása
- Probléma/kérés részletes leírása
 - Pl. hiba reprodukálásához szükséges információ
 - Lehetséges workaround-ok
 - Kipróbált javítási lehetőségek
- Változások történetének tárolása

HIBÁK RÖGZÍTÉSE/HIBAJELENTÉS

- ITS-ek egyik központi eleme a hibák rögzítése
 - Részletes hibaleírás
 - Reprodukálás lépései
 - Becsült erőforrás és határidő rögzítése
 - Priorizálás
 - Felelős hozzárendelés
 - Állapot nyomon követése
- A hiba bejelentés azonban nem egy triviális feladat
 - Incidens menedzsment folyamat definiálja az elemeit
 - ITS-ek csak eszköztámogatást nyújtanak
 - Nem mondják meg, hogyan jelentsünk be egy hibát ☺

INCIDENS MENEDZSMENT

- **Incidens menedzsment (Incident management)** – (IEEE 1044 alapján) Események észlelésének, vizsgálatának, megfelelő tevékenységek elvégzésének és az esemény kiküszöbölésének folyamata.
- Incidensnek nevezünk minden olyan nem tervezett eseményt, ami további vizsgálatokat igényel
- Az incidens menedzsment folyamata magába foglalja:
 - Események feljegyzése
 - Események osztályozása
 - Esemény hatásának beazonosítása
 - Észleléstől a javításig nyomon követhetőség biztosítása
 - Újratesztelés
 - Lezárás
- Nagyon fontos a szervezetek incidens menedzsment folyamatának dokumentálása

INCIDENS JELENTÉS

- Egy jelentés céljai
 - Elegendő információ nyújtása az adott probléma beazonosításához, elkülönítéséhez, kijavításához
 - A teszt vezető képet kap a szoftver minőségéről (hibák száma, prioritása, javítások)
 - Ötleteket adhat a tesztelési folyamat javításához (mely részekre kell fókuszálni?)

INCIDENS JELENTÉS

- Egy jelentésnek általában tartalmaznia kell a következőket
 - Dátum, szerző, jóváhagyó
 - Hatályosság, súlyosság, prioritás
 - Referenciák (teszteset)
 - Elvárt és kapott eredmények
 - Az esemény időpontja
 - A szoftver/rendszer/konfiguráció beazonosítása
 - Szoftver életciklus mely lépésében észleltük?
 - Eltérés leírása
 - Esemény gazdasági hatása
 - Javítás sürgőssége/prioritása
 - Esemény állapota
 - Konklúzió
 - Globális hatások (pl. kapcsolódó szoftverekre)
 - Változások naplózása (change history)

IEEE 829 –TEST INCIDENT REPORT

Heading	
1	Test incident report identifier
2	Summary
3	Incident description
4	Impact

Egyedi azonosító

Rövid leírás: elvárt és kapott eredmények eltérése, érintett magasszintű elemek, incidens kiváltó folyamat leírása

Részletes leírás, ami magában foglalhatja: bemenetek, elvárt eredmények, kapott eredmények, eltérések, dátum és időpont, lépések, környezet, reprodukálási kísérletek, tesztelő megjegyzései, észlelő megjegyzései

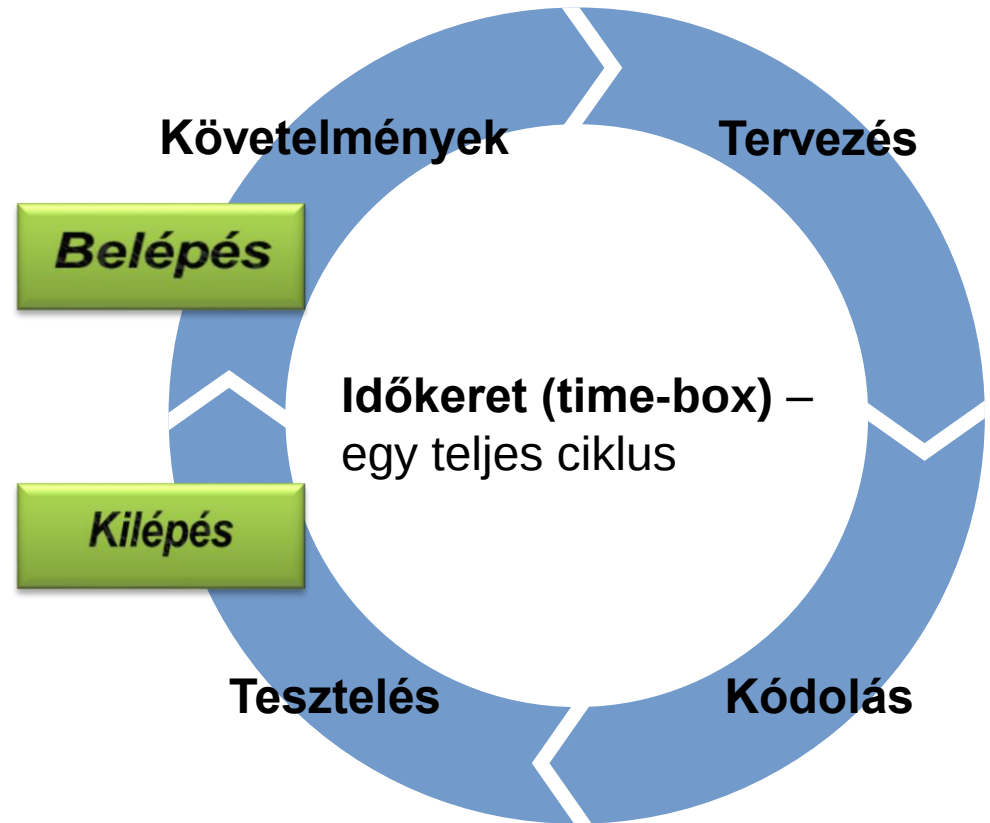
Milyen hatása van az eseménynek a tesztelési folyamatra?

ITS MINT PROJEKTMENEDZSMENT ESZKÖZ

- Jelenlegi ITS-ek számos kiegészítő funkcióval rendelkeznek
 - Tipikusan plug-inekkel tetszőlegesen bővíthetők
- Agilis fejlesztési folyamathoz számos támogatás
 - Scrum, Kanban, stb. támogatás
 - Agile/story board
 - Kanban board
 - Burn-down chart
 - Project back log
 - Sprint áttekintés
 - Story point megjelenítés

ITERATÍV MODELLEK

- Ciklikus modellek
 - A követelményeket nem szükséges rögtön a fejlesztési folyamat legelején véglegesíteni
 - A tervezés, kódolás hamarabb elkezdődhet
 - Ehelyett, az egyes iterációk egyre jobban finomítják a követelményeket, amíg el nem jutunk a végső termékhez
- A fejlesztés végét általában idő és költségkorlátok jelentik

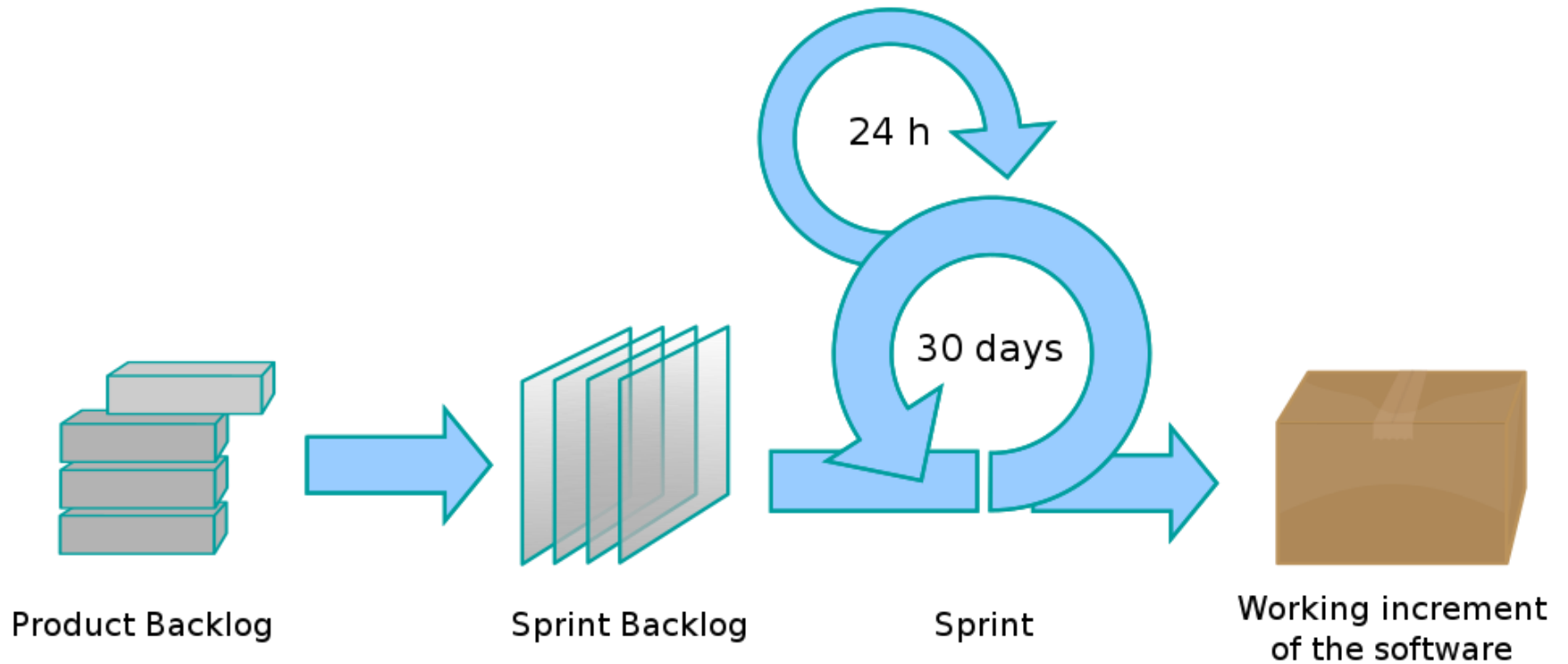


ITERATÍV MODELLEK

- Ebben a folyamatban kulcsszerep jut a felhasználónak
 - Részt vesz a tesztelésben
 - Minden ciklusban újabb változtatásokat eszközölhet, így a végeredmény sokkal inkább megfelel a (menet közben esetleg módosuló) kívánalmainak
- Az ilyen folyamatok hátulütői
 - A formális dokumentáció hiánya megnehezíti a tesztelést
 - Megoldás: teszt alapú fejlesztés (test driven development)
 - A változtatások dokumentálatlansága
 - Megoldás: változáskövető rendszerek használata
 - Nyomonkövethetőség (traceability) hiánya követelmények és végtermék funkciói között
 - Változtatások mennyisége
 - Megoldás: agresszív regressziós tesztelés, változtatásból adódó újabb hibák kiküszöbölésére

- Az iteratív modellek extrém változatai, pl.
 - Scrum
 - Pár hetes ciklusok (sprintek)
 - Napi maximum negyedórás megbeszélések
 - Önszerveződő fejlesztőcsapatok
 - Páros programozás
 - „Scrum board”
 - Kanban
 - Kevésbé szoros folyamat (evolúciós fejlesztés a szigorú release határidők helyett)
 - Nincsenek előre definiált szerepkörök
 - „Kanban board”

SCRUM



SCRUM FOGALMAK

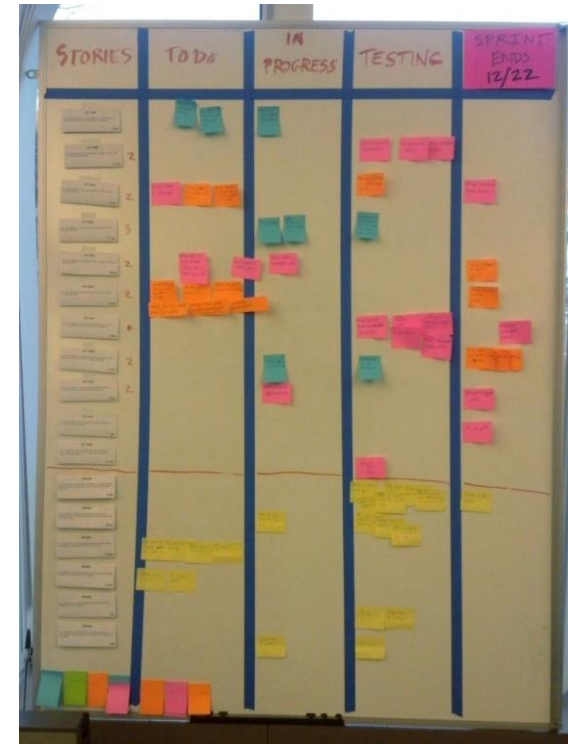
- Keretrendszer, amely magában foglal bizonyos tevékenységeket és meghatározott szerepeket
- Résztvevők csoportosítása: „disznók” és „csirkék”
- Szerepkörök („disznók”)
 - Scrum Master
 - A folyamatot felügyeli és a csapat önálló munkavégzését edzőként segíti
 - Product Owner
 - A projektben érdekelt döntéshozókat képviseli
 - Team Member
 - Csapat, ami a nagyjából 7 főből áll és lefedi az összes munkafolyamatot
- „Csirkék”
 - Felhasználó, üzleti szereplő, menedzser

SCRUM JELLEMZŐK

- **Megbeszélések**
 - Napi „scrum”
 - A futam során naponta tartott megbeszélés
 - Sprint planning
 - Minden futam előtt futamtervező megbeszélés
 - Sprint review
 - Annak áttekintése, hogy mely munkák készültek el és melyek nem
 - Az elkészült munka bemutatása a terméktulajdonos és a fejlesztésben érdekelték részére (demo)
 - Retrospective
 - A csapattagok véleményt alkotnak az elmúlt futamról. A vélemény lehet egy puszta benyomás is, nem kell kidolgozott, szilárd álláspontnak lennie
 - Javaslatokat tesznek a folyamatok továbbfejlesztésére. A javaslatoknak nem kell kiérleltnek lenniük, a kidolgozás nem a visszatekintés része
 - Két kérdés merül fel a megbeszélésen: Mi az, ami jól ment a futam alatt? Mi az, amit a következő futam során jobban lehetne csinálni?

SCRUM KIFEJEZÉSEK

- **Product backlog**
 - a projekt során megvalósításra váró teendők listája, fontossági sorrendben
- **Sprint backlog**
 - konkrét feladatok a következő futamra
- **Burn-down chart**
 - kimutatás a napi eredményekről a futam során
- **Story**
 - a termék funkcióinak magas szintű, megrendelőközpontú leírása
- **Story point**
 - Adott story megvalósításának nehézségét kifejező mérőszám
- **Velocity**
 - A futam során elkészített elemek összpontszáma a csapat sebessége
- **Sprint**
 - a futamtervezés során kiválasztott teendők megvalósítására szánt rövid iteráció (2-4 hét)
- **Scrum task board**
 - A product/print backlog vizuális megjelenítése táblázatos formában



https://hu.wikipedia.org/wiki/Scrum#/media/File:Scrum_task_board.jpg

KANBAN FOGALMAK/JELLEMZŐK

- Folyamatban lévő taskok minimalizálása
 - Context switch overhead minimalizálása
- Evolúciós változtatások
- Ügyfélközpontúság
- Nincsenek előre definiált lépések
- Hat általános elv
 - Vizualizáció (kanban board)
 - WIP korlátozása
 - Folyamat menedzsment
 - Explicit szabályok
 - Feedback loop
 - Kollaboratív, kísérleti evolúció

KANBAN BOARD

Pool of Ideas	Feature Preparation		Feature Selected	User Story Identified	User Story Preparation		User Story Development		Feature Acceptance		Deployment	Delivered
	3 - 10		2 - 5	30	15		15		8		5	
Epic 431	In Progress	Ready			In Progress	Ready	In Progress	Ready (Done)	In Progress	Ready		Epic 294
Epic 478	Epic 444	Epic 662	Epic 602			Story 602-02 Story 602-03	Story 602-06 Story 602-04	Story 602-05 Story 602-01	Epic 401	Epic 609	Epic 694	Epic 386
Epic 562	Epic 589		Epic 302	Story 302-03 Story 302-02	Story 302-01 Story 302-06	Story 302-07 Story 302-08	Story 302-09	Story 303-05 Story 302-04	Epic 468	Epic 577	Epic 276	Epic 419
Epic 439	Epic 651		Epic 335	Story 335-09 Story 335-08	Story 335-10 Story 335-01 Story 335-03	Story 335-05 Story 335-02	Story 335-06 Story 335-07		Epic 362		Epic 339	Epic 388
Epic 329			Epic 512	Story 512-04 Story 512-05	Story 512-07 Story 512-06 Story 512-03	Story 512-01					Epic 521	Epic 287
Epic 287											Epic 582	Epic 274
Epic 606	Discarded											
	Epic 511	Epic 213										
	Epic 221											

Policy

Business case showing value, cost of delay, size estimate and design outline.

Policy

Selection at Replenishment meeting chaired by Product Director.

Policy

Small, well-understood, testable, agreed with PD & Team

Policy

As per "Definition of Done" (see...)

Policy

Risk assessed per Continuous Deployment policy (see...)

KONKRÉT ITS ESZKÖZ

- Tipikusan webes alkalmazások egy adatbázis backend szerver komponenssel
- Webes UI elérés + API
- Ingyenes vs. fizetős, például
 - Bugzilla
 - IBM ClearQuest
 - HP Quality Center
 - JIRA
 - Redmine
 - Team Foundation Server
 - Trac



- <http://www.redmine.org/>
- Ingyenes, nyílt forrású, webes projekt menedzsment eszköz
- Projekt/alprojekt/feladat hierarchia
- WiKi, fórum, ráfordítás követés
- Beépített Gantt diagram nézet
- VCS integráció, repository browser, diff nézegető
- Ruby on Rails
 - Több platformot is támogat
 - Tetszőleges RDBMS köthető alá
- Az egyik legnépszerűbb nyílt forrású eszköz

REDMINE FUNKCIÓK

- Több projekt követése
- Rugalmas, szerepkör alapú hozzáférés vezérlés
- Teljes körű ITS
- Gantt és naptár nézet
- Integrált dokumentum kezelő
- Projektenkénti WiKi + fórum
- Idő ráfordítás követés
- VCS integráció (SVN, Git, Mercurial, Bazaar, Darcs)
- LDAP autentikáció támogatás
- 34 nyelv és sok adatbázis típus támogatása
- Plug-in bővíthetőség
- REST API

REDMINE SCREENSHOT

[Home](#)
[My page](#)
[Projects](#)
[Administration](#)
[Help](#)

[Logged in as admin](#)
[My account](#)
[Sign out](#)

git-helloworld

[Search:](#)

[Overview](#)
[Activity](#)
[Issues](#)
[New issue](#)
[News](#)
[Documents](#)
[Files](#)
[Repository](#)
[Settings](#)

New issue

Tracker *

Subject *

Description

B

I

U

S

C

H1

H2

H3

☰

☷

📎

📎

pre

📎

📎

Text formatting: [Help](#)

Status *

Start

Priority *

Due date

Assigned to

Estimated time Hours

% Done

Files

Browse...

Optional description

REDMINE PLUG-IN STRUKTÚRA

- Kiegészítő funkciók hozzáadása az alap Redmine működéshez
- Központi tárhely
 - <http://www.redmine.org/plugins>
- Saját kiegészítők is írhatók
- Bármilyen téren kiegészíthetik a Redmine-t
 - Kanban Tool
 - Diff Popup
 - Agile
 - Easy Gantt
 - Advanced reminder
 - Stb., stb., stb.

REDMINE AGILE PLUG-IN

The screenshot displays the Redmine Agile plugin demo interface. The browser window title is "Agile board - Agile board - Redmine Agile plugin demo - RedmineCRM demo". The top navigation bar includes links for Home, My page, Projects, Contacts, Deals, Finance, People, Administration, Help & Support, and a login status for "admin". A search bar is present with the text "Redmine Agile plugin demo". The main navigation bar includes Overview, Activity, Roadmap, Issues, New issue, Gantt, Agile (selected), Wiki, Contacts, and Settings.

The main content area is titled "Agile board" and features three columns: "New (5)", "In Progress (4)", and "Closed (5)".

- New (5):** Contains five user stories: #586 (Cumulative flow chart), #589 (Velocity chart), #580 (Add big avatars), #583 (Print Tasks), and #579 (Assign Task to Area).
- In Progress (4):** Contains four items: User story #588 (Backlog and sprint, assigned to Wozniak Stephen), Task #584 (Setup CI server, assigned to Pong Steven), User story #587 (Sortable board cards, assigned to Stone Elizabeth), and another instance of User story #587 (assigned to Stone Elizabeth).
- Closed (5):** Contains five items: User story #581, User story #585 (Issues burndown chart, assigned to Wozniak Stephen), User story #582 (Done/Undone Task, assigned to Pong Steven), Task #591 (PDF export), and another instance of User story #581.

A tooltip for "User story #578" is visible, stating: "Edit Project with many lines. How will it look like on the board".

On the right side, there are sections for "Members" (listing Smith Paul, Pong Steven, Kutcher Ashton, Stone Elizabeth, and Wozniak Stephen), "Issues" (with links for View all issues, Summary, Gantt, and Agile board), "Agile charts" (with links for Issues burndown, Created/Closed, Velocity, Lead time, and Trackers cumulative flow), and "My agile boards" (with a link for User stories board).



- <https://www.atlassian.com/software/jira>
- Atlassian fejleszti
- Integrált agilis fejlesztés támogató eszközök
- Teljes körű ITS
- Egy bizonyos rangsorolás szerint ez a legnépszerűbb ITS
 - <https://project-management.zone/ranking/category/issue>
- Három különálló csomagból tevődik össze
 - Jira Software
 - Az alap szoftver agilis támogatással
 - Jira Core
 - Általános projekt menedzsment funkciók
 - Jira Service Desk
 - IT és üzlet service desk-ek számára kifejlesztett komponens

JIRA REFERENCIA

- Főbb felhasználók
 - Fedora
 - Hibernate
 - Twitter
 - Skype
 - NASA
 - US DoD
 - Apache Software Foundation

JIRA SCREENSHOT

The screenshot displays the Jira interface for the 'Teams in Space' project. The left sidebar contains navigation links: Backlog, Board (selected), Reports, Releases, Components, Issues, Repository, Add item, and Settings. The main area shows a Kanban board with four columns: TO DO (5 items), IN PROGRESS (5 items), CODE REVIEW (2 items), and DONE (8 items). Each item card includes a title, a team label, a status icon, a count, a due date, and an assignee.

Board Release ...

TO DO 5

- Engage Jupiter Express for outer solar system travel
SPACE TRAVEL PARTNERS
✓ ↑ 5 TIS-25
- Create 90 day plans for all departments in the Mars Office
LOCAL MARS OFFICE
+ ⊘ 9 TIS-12
- Engage Saturn's Rings Resort as a preferred provider
SPACE TRAVEL PARTNERS
+ ↑ 3 TIS-17
- Enable Speedy SpaceCraft as the preferred
SPACE TRAVEL PARTNERS

IN PROGRESS 5

- Requesting available flights is now taking > 5 seconds
SEESPACEEZ PLUS
+ ↑ 3 TIS-8
- Engage Saturn Shuttle Lines for group tours
SPACE TRAVEL PARTNERS
✓ ↑ 4 TIS-15
- Establish a catering vendor to provide meal service
LOCAL MARS OFFICE
+ ↑ 4 TIS-15
- Engage Saturn Shuttle Lines for group tours
SPACE TRAVEL PARTNERS

CODE REVIEW 2

- Register with the Mars Ministry of Revenue
LOCAL MARS OFFICE
+ ↑ 3 TIS-11
- Draft network plan for Mars Office
LOCAL MARS OFFICE
✓ ↑ 3 TIS-15

DONE 8

- Homepage footer uses an inline style - should use a class
LARGE TEAM SUPPORT
+ ↑ TIS-68
- Engage JetShuttle SpaceWays for travel
SPACE TRAVEL PARTNERS
+ ↑ 5 TIS-23
- Engage Saturn Shuttle Lines for group tours
SPACE TRAVEL PARTNERS
✓ ↑ TIS-15
- Establish a catering vendor to provide meal service
LOCAL MARS OFFICE

- <https://trac.edgewall.org/>
- Nyílt forrású, ingyenes
- Python alapú
- Erős WiKi funkciók
- Teljes körű ITS
- Plug-inek segítségével bővíthető
- SVN és Git integráció
 - Plug-inekkel szinte bármely más VCS-el integrálható
- Testre szabható riportok kérése tárolt SQL-ek segítségével
- RSS támogatás



- Főbb felhasználók
 - Django
 - Ffmpeg
 - jQuery UI
 - OpenStreetMap
 - WebKit
 - WordPress

TRAC SCREENSHOT



logged in as cboos | [Logout](#) | [Settings](#) | [Help/Guide](#) | [About Trac](#)

[Wiki](#) | [Timeline](#) | [Roadmap](#) | **[Browse Source](#)** | [View Tickets](#) | [New Ticket](#) | [Search](#) | [Admin](#)

[Revision Log](#)

root / [trac](#) / **versioncontrol**

Browse branch: Jump to tag: View revision:

Name ▲	Size	Rev	Age	Last Change
↑ ../				
tests		12:ef52d6095ff7	4 days	Administrator@localhost: Big dos2unix conversion. Multiple consecutive CRLFs apparently confuses ...
web_ui		24:61adfa325d	11 hours	Christian Boos <cboos@neuf.fr>: Refactored changeset event provider logic in Repository.get_changesets ...
__init__.py	38 bytes	12:ef52d6095ff7	4 days	Administrator@localhost: Big dos2unix conversion. Multiple consecutive CRLFs apparently confuses ...
api.py	10.1 kB	24:61adfa325d	11 hours	Christian Boos <cboos@neuf.fr>: Refactored changeset event provider logic in Repository.get_changesets ...
cache.py	5.8 kB	12:ef52d6095ff7	4 days	Administrator@localhost: Big dos2unix conversion. Multiple consecutive CRLFs apparently confuses ...
diff.py	10.8 kB	12:ef52d6095ff7	4 days	Administrator@localhost: Big dos2unix conversion. Multiple consecutive CRLFs apparently confuses ...
hg.py	19.6 kB	30:1bc1869bd165	7 hours	Christian Boos <cboos@neuf.fr>: Implemented revision log for directories.
svn_authz.py	3.6 kB	12:ef52d6095ff7	4 days	Administrator@localhost: Big dos2unix conversion. Multiple consecutive CRLFs apparently confuses ...
svn_fs.py	18.6 kB	14:191327139aaa	3 days	Administrator@localhost: The versioncontrol.web_ui modules are now abstract components. ...

Note: See [TracBrowser](#) for help on using the browser.



Powered by **Trac 0.9b2**
By Edgewall Software.

Visit the Trac open source project at
<http://trac.edgewall.com/>

GITLAB ITS

 GitLab

Projects ▾

Groups

Activity

Milestones

Snippets

 GitLab Community Edition

Overview

Repository

Issues

9,082

List

Boards

Labels

Service Desk

Milestones

Merge Requests

478

CI / CD

Snippets

Members

Collapse sidebar

GitLab.org > GitLab Community Edition... > Issues

Open 9,082 Closed 19,215 All 28,297

Search or filter results...

Last created ▾

something went wrong on our end

#38391 · opened 18 minutes ago by arunkumar.nagarajan

updated 18 minutes ago

Add backend check to project deletion modal

#38390 · opened 26 minutes ago by Winnie Hellmann

Platform backend

updated 26 minutes ago

MR without CI cannot be accepted

#38389 · opened 47 minutes ago by Cédric Tabin

updated 47 minutes ago

The Repository > Graph get 404 when the target branch contains "/" sign

#38388 · opened about an hour ago by Ferenczi Krisztián

updated about an hour ago

Docker registry push hangs with dash in image name

#38387 · opened about an hour ago by Martin Bertschler

updated about an hour ago

Changing primary email address sends "thanks for signing up" email

#38386 · opened about an hour ago by Mike Greiling

Platform bug emails notifications settings

updated about an hour ago

GPG tooltips not working in Safari

#38385 · opened about an hour ago by Mike Greiling

Platform bug frontend gpg

updated about an hour ago

Error 503 when commit any file in UI

#38384 · opened about 2 hours ago by Иван Буймов

updated about 2 hours ago

Add projects to user

0 of 3 tasks completed

#38382 · opened about 3 hours ago by Christopher Linn

updated about 3 hours ago

Can't choose subgroup while forking project

#38381 · opened about 4 hours ago by Andres Rovira

updated about 4 hours ago

Smoke/Load tests in production

0 of 3 tasks completed

#38380 · opened about 4 hours ago by Joshua Lambert

Prometheus direction feature proposal

updated about 4 hours ago

TestEnv can't handle component versions with a branch name in CI

#38378 · opened about 6 hours ago by Alejandro Rodríguez

bug

1 updated about 6 hours ago

Instruction to: "git remote add origin" fails when existing repo already has an origin

#38377 · opened about 8 hours ago by John Ellison

updated about 7 hours ago

Add note for upgrading Helm chart prior to 1.3.5 - docs

#38374 · opened about 9 hours ago by Joshua Lambert

Documentation

1 updated about 9 hours ago

GITHUB ITS

IssuesPull requestsLabelsMilestones

Filters

New issue

104 Open ✓ 9,660 Closed

AuthorLabelsMilestonesAssigneeSort

!

.form-group-sm .form-group-lg shrink textarea

confirmedcss

4

#13989 opened 11 hours ago by limitstudios v3.2.1

!

Tooltip unnecessarily breaks into multiple lines when positioned to the right

confirmedjs

0

#13987 opened 15 hours ago by hnrch02 v3.2.1

!

Tooltip Arrows in Modal example facing wrong way

css

6

#13981 opened a day ago by SDCore

!

Table improvement

css

0

#13978 opened a day ago by Tjoosten

!

docs/dist files

docs

7

#13977 opened 2 days ago by XhmikosR v3.2.1

!

Potential solution to #4647

js

4

#13976 opened 2 days ago by julioarmandof

!

Bootstrap site: right-hand navigation text becomes rasterized after scrolling

cssdocs

4

#13974 opened 2 days ago by mg1075 v3.2.1

!

Dropdown toggle requires two clicks

js

1

#13972 opened 2 days ago by Kizmar

ITS-EK ÖSSZEHASONLÍTÁSA

- https://en.wikipedia.org/wiki/Comparison_of_issue-tracking_systems
- Készítő
- Licenz
- Implementáció nyelve
- Back end
- Kiadás dátuma

KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

SZOLGÁLTATÁS ORIENTÁLT TECHNIKÁK – SOA, REST

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap

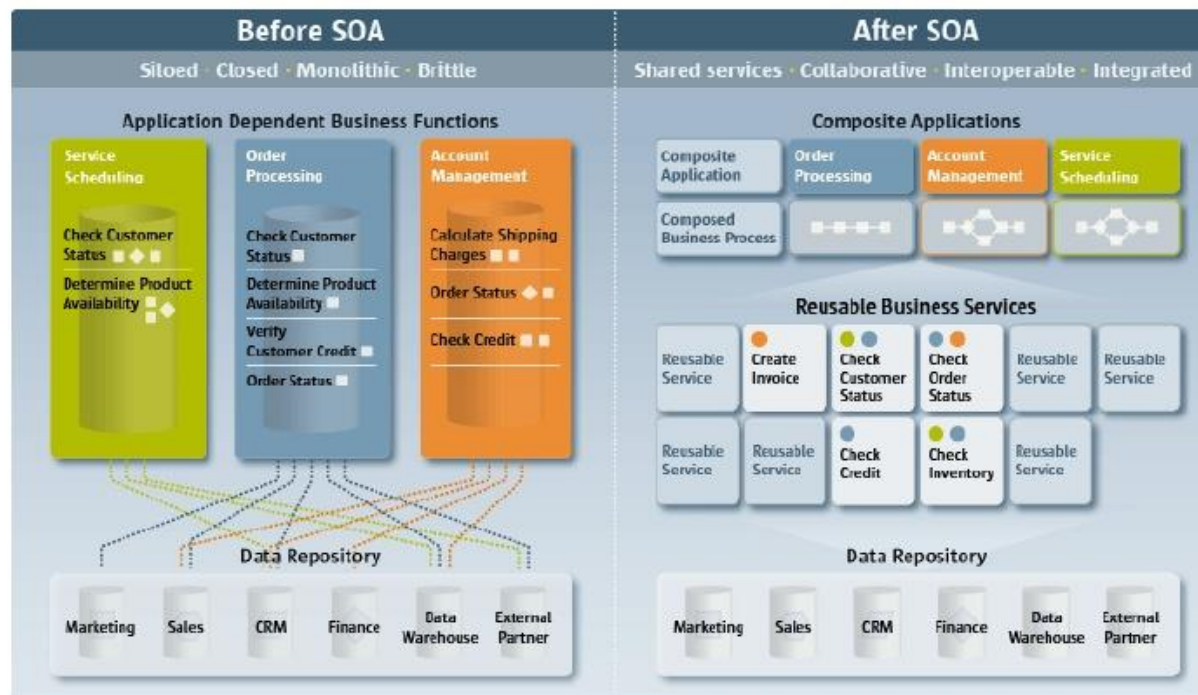


BEFEKTETÉS A JÖVŐBE

- Service-oriented architecture
- Alapvető építőelemek a szolgáltatások
 - Egy-egy üzleti folyamatnak felelnek meg
 - Laza kapcsolat és együttműködés
 - Szabványosított komponensek
 - Biztonsági szempontok integrálása
 - Újra hasznosíthatók
 - Újra kombinálhatók
- Megfelelő mechanizmus különböző működési modellek meghatározására
- Alkalmas különböző rendszerek integrációjához is
 - A szolgáltatások jól definiált, általános interfészként működhetnek

SOA ELŐTT/UTÁN

Before & After SOA



- A szolgáltató és a szolgáltatást igénybe vevő a hálózat segítségével kerül kapcsolatba egymással
 - Jellemzően két teljesen idegen rendszer elemét képezik
- A kapcsolatépítés a következő folyamatokból tevődik össze
 - A szolgáltatás közzéteszi magát egy szolgáltatásokat leíró nyilvánosan elérhető adatbázisban
 - A szolgáltatást igénylő felkeresi az említett adatbázisból a meghívni kívánt szolgáltatást, és kap egy szolgáltatás-leíró, mely tartalmazza a szolgáltatás lényeges adatait
 - hogyan lehet meghívni, milyen típusú és mennyi paramétert vár, milyen típusú a visszatérési értéke, stb.
 - Az igénylő a leíró segítségével meghívja a szolgáltatást a szükséges adatok átküldésével
 - A szolgáltatás feldolgozza a kérést, és visszaküldi a feldolgozás eredményét az igénylőnek

SOA TÁGABB ÉRTELMEZÉSBEN

- Egy nagyobb rendszert kell elképzelnünk, mely
 - Lazán csatolt együttműködő szolgáltatásokból épül fel
 - Ezek a szolgáltatások közös formális nyelven kommunikálnak egymással, mely nyelvfüggetlen az adott szolgáltatás implementációs nyelvétől és környezetétől
 - A szolgáltatások interfész-definíciója elrejtí az architektúra elől a nyelvfüggő implementációs részeket
 - A SOA szabványosított komponensekből felépülő rendszer, amelynek egyik legfontosabb eleme egy kommunikációs réteg, mely rétegen keresztül válnak elérhetővé a háttérrendszerekben azonosított szolgáltatások, melyeket a szolgáltatás-interfészen keresztül érünk el

SOA ELŐNYÖK/HÁTRÁNYOK

- Előnyei

- Komponensalapú építkezés, mely lehetővé teszi a komponensek újra felhasználhatóságát
- Leváltja a monolitikus (silóorientált), nehezen átlátható és kezelhető rendszereket, így a kritikus helyek könnyebben beazonosíthatók, karbantarthatók
- Rugalmas és hatékony alkalmazáskörnyezet és architektúra
- Elosztottság támogatása: a felhasználói felület elválik az implementációs rétegtől
- Könnyen párhuzamosítható fejlesztési feladatok, mely szintén az elosztottságból fakad

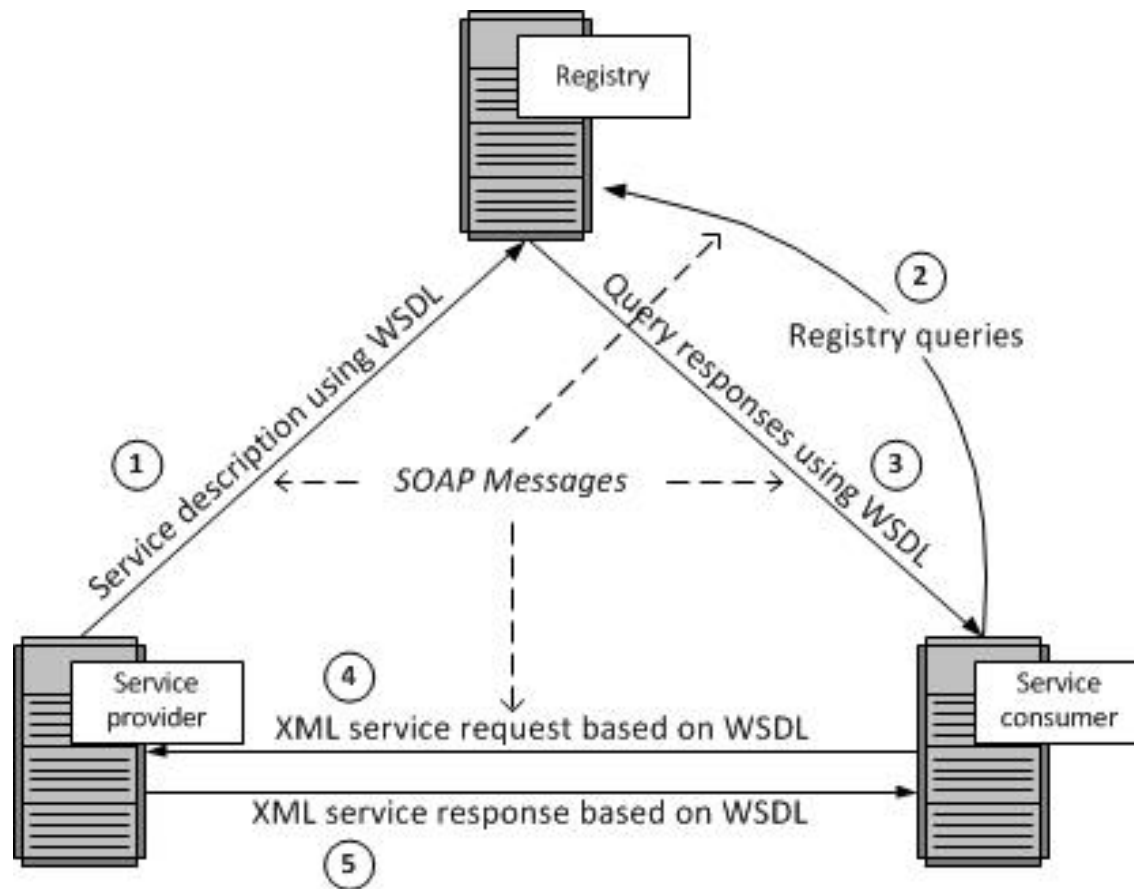
- Hátrányok

- A rugalmasság biztosítása nagyobb erőforrásigényekkel jár
 - A jelenlegi technológiai lehetőségek azonban megfelelnek ezen igényeknek

SOA TIPIKUS MEGVALÓSÍTÁSA

- A valóságban általában webszolgáltatások (WebService)
- UDDI
 - Universal Description, Discovery, and Integration
 - „Adatbázis”, ahová a szolgáltatások regisztrálásra kerülnek
 - Platformfüggetlen, XML-alapú nyilvántartó rendszer
- WSDL
 - Szolgáltatás leíró nyelv (XML)
 - UDDI ezt használja
 - Szolgáltatás végpontja, neve, paraméterei, típusok, stb.
- SOAP
 - Simple Object Access Protocol
 - XML alapú kommunikációs protokoll webszolgáltatásokhoz
 - HTTP vagy egyéb hálózati protokollon továbbítódnak

SOA TIPIKUS MEGVALÓSÍTÁSA



WSDL PÉLDA

```
<definitions
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://ws.mkyong.com/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  targetNamespace="http://ws.mkyong.com/"
  name="HelloWorldImplService">
  <types>
  <xsd:schema>
    <xsd:import namespace="http://ws.mkyong.com/"
      schemaLocation="http://localhost:9999/ws/hello?xsd=1"/></xsd:import>
  </xsd:schema>
  </types>

  <message name="getHelloWorldAsString">
    <part name="parameters" element="tns:getHelloWorldAsString"/></part>
  </message>
  <message name="getHelloWorldAsStringResponse">
    <part name="parameters" element="tns:getHelloWorldAsStringResponse"/></part>
  </message>

  <portType name="HelloWorld">
    <operation name="getHelloWorldAsString">
      <input message="tns:getHelloWorldAsString"/></input>
      <output message="tns:getHelloWorldAsStringResponse"/></output>
    </operation>
  </portType>
```

```
<binding name="HelloWorldImplPortBinding" type="tns:HelloWorld">

  <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document">
  </soap:binding>
  <operation name="getHelloWorldAsString">
    <soap:operation soapAction=""></soap:operation>
    <input>
      <soap:body use="literal"/></soap:body>
    </input>
    <output>
      <soap:body use="literal"/></soap:body>
    </output>
  </operation>

</binding>

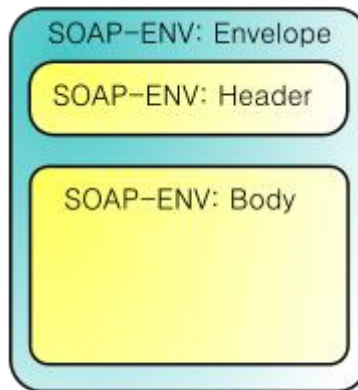
<service name="HelloWorldImplService">

  <port name="HelloWorldImplPort" binding="tns:HelloWorldImplPortBinding">

    <soap:address location="http://localhost:9999/ws/hello"/></soap:address>

  </port>
</service>
</definitions>
```

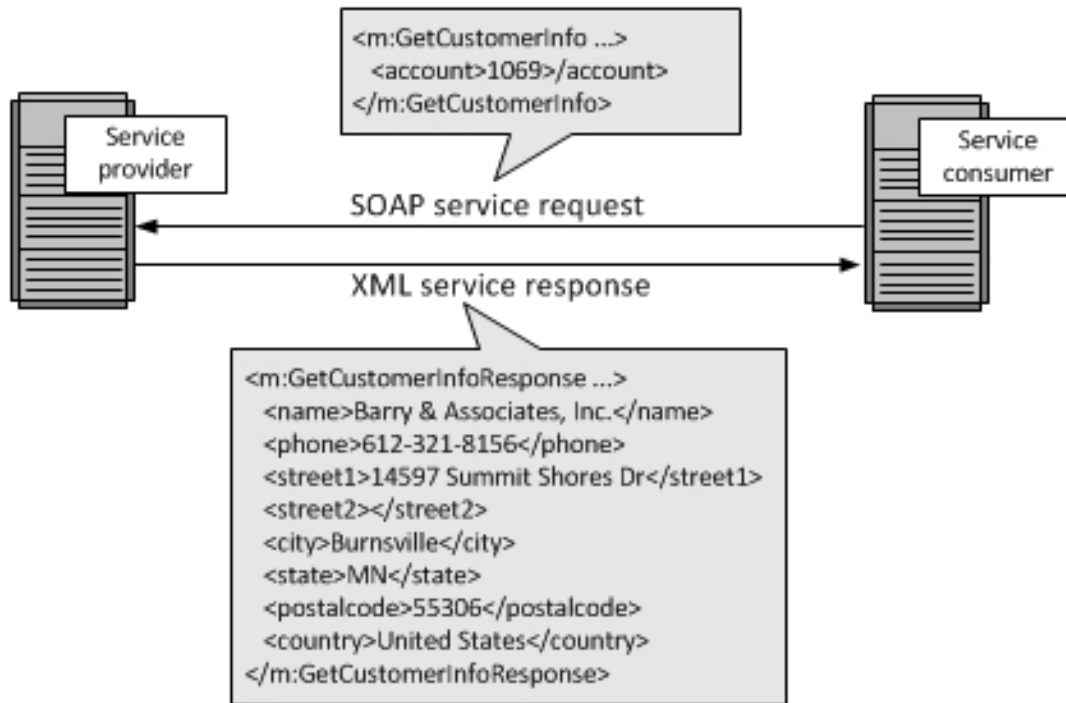
SOAP ÜZENET PÉLDA



<https://hu.wikipedia.org/wiki/SOAP#/media/File:SOAP.svg>

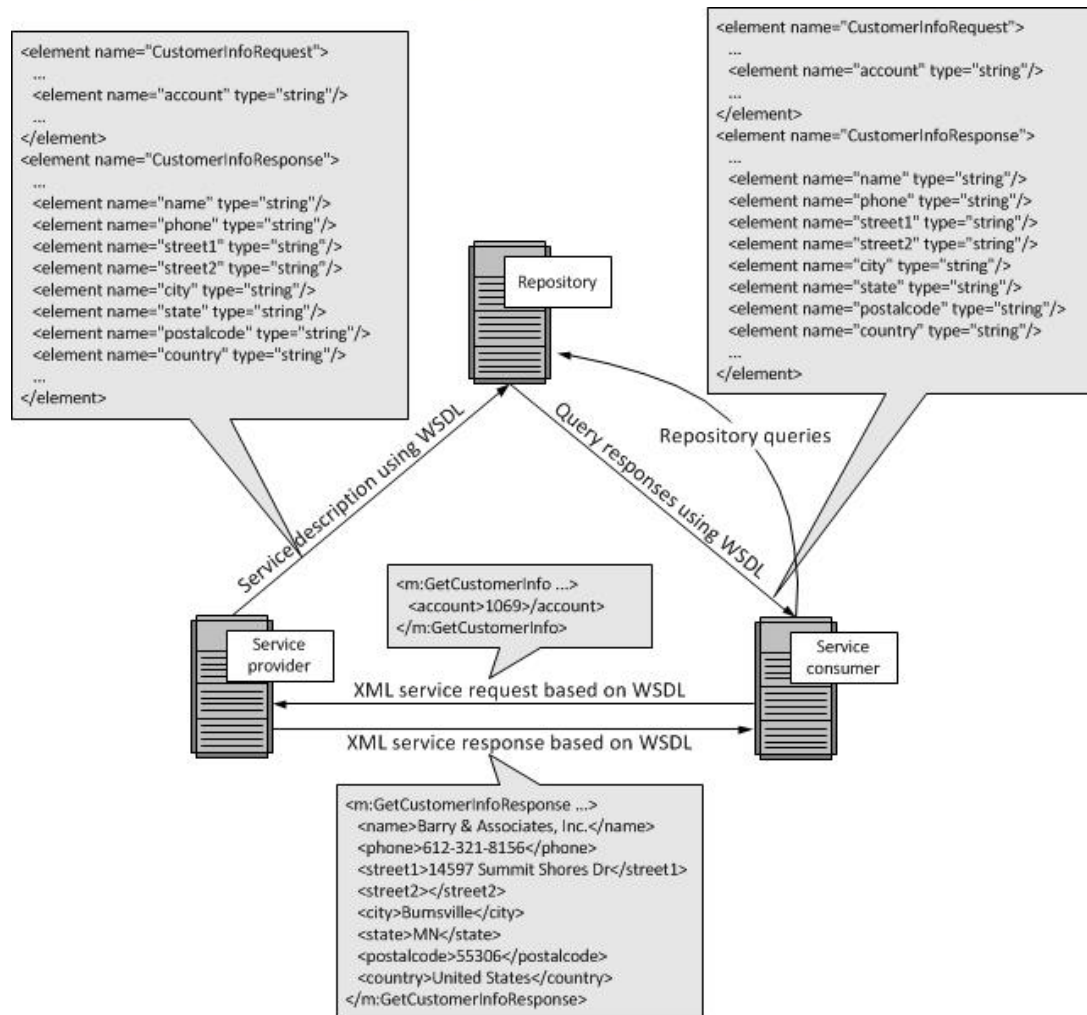
```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

SOAP ALAPÚ KOMMUNIKÁCIÓ



https://www.service-architecture.com/images/web_services/soap_messages.jpg

SOA MEGVALÓSÍTÁSA SOAP ÜZENETEKKEK



SOAP TÁMOGATÁS JAVA-BAN

- SOAP alapú webszolgáltatások készítése/felhasználása különböző szinteken támogatott
- JAX-WS
 - Java API for XML Web Services
- Java EE 6
 - JSR 224-es szabvány
- Szerver/kliens oldali kód
 - Annotációk
 - Kód generáló eszközök
 - WSDL támogatás
- Csak API, implementációk
 - Apache Axis2
 - Metro
 - Stb.

SOAP BIZTONSÁG

- Biztonsági megfontolások szerves részét képezik a protokollnak
 - Nem csak az átvivő közeg biztonságára épít (pl. TLS, SSH)
- WS-Security
 - Végpont szintű titkosítás
 - Aláírás integritás és letagadhatatlanság védelemhez
 - XML titkosítás
- Teljesítmény csökkenéssel járhat

REST

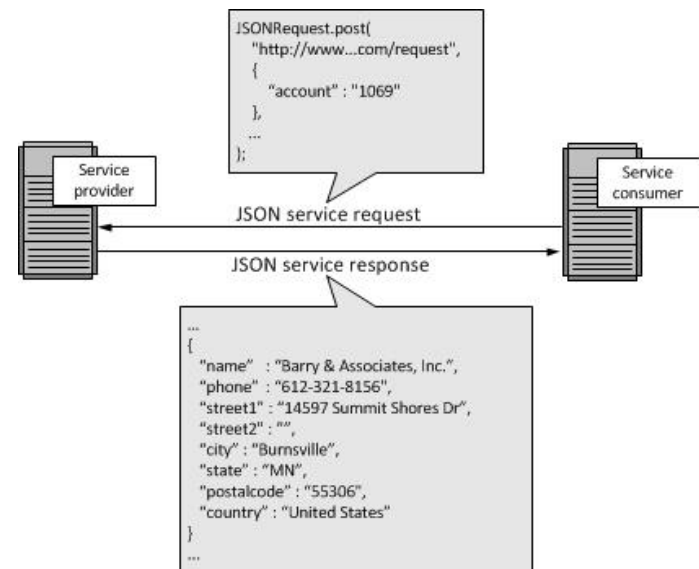
- Representational State Transfer
 - Egy kevésbé kötött SOA megoldás
- HTTP alapú, egyéb megkötés nincs
- Valami „RESTful”, ha eleget tesz a következőnek
 - Minden komponens jól definiált interfészeken keresztül kommunikál
 - Minden komponens egyértelműen azonosítható egy URL-lel
 - Kliens/szerver architektúrát követ
 - Minden kommunikáció állapot mentes
 - Réteges architektúrát követ, és az adatok bármely rétegben cache-elhetők

REST ALAPÚ WEBSZOLGÁLTATÁS

- Szolgáltatások azonosítása URL segítségével (HTTP 1.1 verbs)
 - GET
 - POST
 - PUT
 - DELETE
- Tetszőleges formátumú lehet a válasz, nem csak XML
 - CSV
 - JSON
 - RSS

REST

- Összegezve
 - Technológia és platform független
 - Lazán csatolt komponensek kommunikációja
 - Standard web protokollok feletti interfészek
 - Nincs szükség formális szerződésre a szolgáltató és fogyasztó között



REST PÉLDA

- [Weather API](#)
- Példa szolgáltatás URL
 - http://api.wunderground.com/api/Your_Key/conditions/q/CA/San_Francisco.json
- Your_Key
 - Általában szükséges egy access token
- <http://api.wunderground.com/api/3afa5ad5c38e14e4/conditions/q/Szeged.json>
- A meghívandó webszolgáltatást az URL írja le
 - Szeged időjárására vagyok kíváncsi
 - JSON formátumban
 - .xml -> XML formátumban adja vissza ugyanazt az információt

REST WEBSZOLGÁLTATÁS FELHASZNÁLÁSA

- Ahogy az előző példán láttuk, többnyire egy URL kell
- Akár egy böngésző vagy más standard HTTP request/response kezelésére alkalmas eszköz alkalmas a webszolgáltatás meghívására
- Amennyiben POST kérést küldünk (pl. sok paraméter miatt)
 - Böngésző önmagában már nem ideális
 - Plug-in-ek segítenek (pl. Postman, RESTClient, stb.)
- Bármelyik programnyelven könnyen készíthető consumer kód

REST WEBSZOLGÁLTATÁS KÉSZÍTÉSE

- Standard web server implementáció
- Pl. servlet Java esetén
- A szolgáltatásokat az egyes URL-ek határozzák meg
 - Valamint a HTTP verb-ek (GET, POST, stb.)
- Attól függően hogy milyen URL-re milyen HTTP kérés érkezett, a megfelelő üzleti folyamat fut le
 - Az eredmény tetszőleges formában visszaküldhető HTTP response-ban
- Számos keretrendszer támogatás
 - Pl. Spring IO Java esetén

REST WEBSZOLGÁLTATÁS KÉSZÍTÉSE

- SOAP hívás

```
1 <SOAP-ENV:Envelope xmlns:SOAP
2   ENV="http://schemas.xmlsoap.org/soap/envelope/">
3   <SOAP-ENV:Header>
4     some data here...
5   </SOAP-ENV:Header>
6   <SOAP-ENV:Body>
7     <GetBenefits>
8       <user>123-45-6789</user>
9       <type>full_time_employee</type>
10    </GetBenefits>
11  </SOAP-ENV:Body>
12 </SOAP-ENV:Envelope>
```

- REST hívás

- <http://humanresources.com/benefits?user='123-45-6789'&type='full time employee'>

- Java servlet implementáció

```
1 protected void doPost( HttpServletRequest req, HttpServletResponse resp
2   throws ServletException, IOException
3 {
4   ServletOutputStream out = resp.getOutputStream();
5   String response;
6
7   String userSSID = req.getParameter("user");
8   String userType = req.getParameter("type");
9   if ( userType.equals("full_time_employee")) {
10     Employee emp = lookupUser(userSSID);
11     String medPlan = emp.getMedicalPlan();
12     String dntPlan = emp.getDentalPlan();
13     String retPlan = emp.getRetirementPlan();
14     Response = "User " + emp.getFullName() +
15               " has medical plan: " + medPlan +
16               ", and dental plan: " + dntPlan +
17               ", and retirement plan: " + retPlan;
18   }
19   else {
20     // ...
21   }
22   // Output the response from the worker
23   out.println(response);
24 }
```

REST WEBSZOLGÁLTATÁS KÉSZÍTÉSE

- Java esetén például
 - Spring IO
 - Spring Boot
- Konfiguráció és XML nélküli stand-alone alkalmazások fejlesztése
- Beágyazott alkalmazás szerver
- Annotációk használata
 - RestController
 - RequestMapping
 - Stb.

SPRING REST

```
package hello;

import java.util.concurrent.atomic.AtomicLong;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

@RestController
public class GreetingController {

    private static final String template = "Hello, %s!";
    private final AtomicLong counter = new AtomicLong();

    @RequestMapping("/greeting")
    public Greeting greeting(@RequestParam(value="name", defaultValue="World")
        return new Greeting(counter.incrementAndGet(),
            String.format(template, name));
    }
}
```

REST BIZTONSÁG

- Alapvetően az átviteli közeg biztonságára épít
 - Mivel a kommunikáció HTTP-n keresztül megy, TLS biztosítja a titkosítást
- Azonosításhoz szintén HTTP módszereket használhatunk
 - Basic authentication
 - Digest authentication
- Pont-pont alapú titkosítás nincs integrálva
 - Ha szeretnén üzenet szintű titkosítást, az magunknak kell implementálni

REST VS SOAP

- SOAP

- Nehézszúlyú megoldás az alábbi előnyökkel
 - Független az átviteli protokolltól (REST csak HTTP-n mehet)
 - Jól működik elosztott környezetben (REST pont-pont kommunikációt feltételez)
 - Szabványosított
 - Bővíthetőség
 - Beépített hibakezelés
 - Bizonyos nyelvekkel használva jól automatizált

REST VS SOAP

- REST

- Általánosságban könnyebb felhasználás és rugalmasság az alábbi előnyökkel
 - Nincs szükség speciális/drága eszközökre a webszolgáltatások eléréséhez (sima HTTP kérés)
 - Gyorsabb elsajátíthatóság
 - Hatékony (SOAP mindenhez XML-t használ, REST esetében alkalmazhatunk más formátumot, kisebb csomagokat)
 - Gyors (nincs szükség komoly feldolgozásra)
 - Konceptcionálisan közelebb áll az egyéb web technológiákhoz

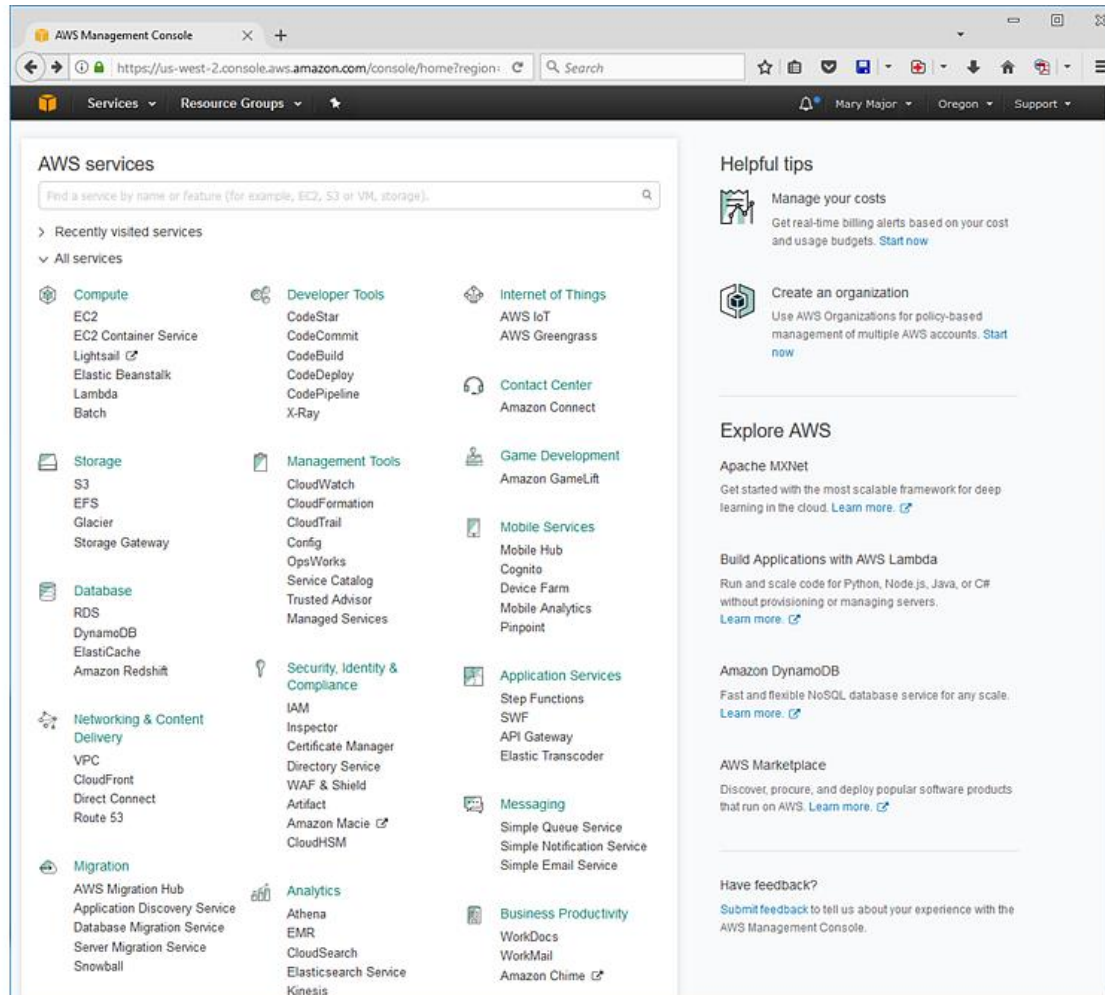
REST VS SOAP, MELYIKET VÁLASSZAM?

- Attól függ... 😊
- Általánosságban
 - SOAP
 - M2M kommunikáció
 - Ahol fontos az átvitel biztonsága és az adatok integrációja
 - Ahol fontos, hogy a szolgáltató és felhasználó közötti kommunikációhoz legyen „formális szerződés”
 - Ahol fontos az állapot megőrzése
 - REST
 - Publikus API-k megvalósításához
 - Ahol számít a webszolgáltatás válaszának feldolgozási ideje (pl. JSON gyorsabb, mint az XML)
 - Ahol gyors válaszra, sok kommunikációra van szükség (pl. mobil appok)

WEBSZOLGÁLTATÁSOK A FELHŐBEN

- Cloud szolgáltatók webszolgáltatásokat nyújtanak
 - Fejlesztők számára
 - Általános feladatok webszolgáltatásként elérhetők
 - Gépi tanulás
 - Tárolás
 - Mobil szolgáltatások
 - Adatbázisok
 - Stb.
 - Például
 - Amazon AWS
 - Microsoft Azure
- Saját webszolgáltatások is készíthetők
 - Amelyeket a felhőben üzemeltetünk
 - Bárki elérheti és használhatja

AWS MANAGEMENT CONSOLE



<http://docs.aws.amazon.com/awsconsolehelpdocs/latest/gsg/images/console-home.png>

KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

JAVA TÉMÁK AZ ALAPOKON TÚL

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

JAVA REFLECTION

- Egy Java technika
- Lehetővé teszi az osztályok, interfészek, mezők és metódusok futás időben történő vizsgálatát
 - Nem kell tudnunk az adott elem nevét fordítási időben
 - Példányosíthatunk objektumot, meghívhatunk metódust, lekérdezhetünk/beállíthatunk mező értéket
 - Mindezt akár úgy, hogy az adott elem neve egy dinamikusan összeállított string-ből jön
- Hasznos többek között például
 - JavaBean-ek feldolgozásához
 - Annotációk kezeléséhez
 - Pl. JUnit tesztek

CLASS OBJECT

- Létezik egy speciális típus, Class
 - Minden Java objektum egy megfelelő class-hoz tartozik
 - `getClass()` metódus adja vissza (Object-ben definiált)
 - Vagy az osztály `class` adattagja (pl. `Foo.class`)
- Adott objektum osztályáról tartalmaz meta-adatokat
 - Név
 - Csomag
 - Metódusok
 - Mezők
 - Konstruktorkok
 - Annotációk
- A reflection API részét képezi

```
public void process(Object obj) {  
    if (obj.getClass().isAnnotationPresent(Processable.class)) {  
        // process somehow;  
    }  
}
```

JAVA REFLECTION API

- A `java.lang.reflect` csomag tartalmazza
- Főbb inetrfészek
 - Member
 - Type
- Főbb osztályok
 - Method
 - Field
 - Constructor
- Részletes dokumentáció
 - <https://docs.oracle.com/javase/7/docs/api/java/lang/reflect/package-summary.html>

JAVA REFLECTION API PÉLDA

```
import java.lang.reflect.*;

public class DumpMethods {
    public static void main(String args[])
    {
        try {
            Class c = Class.forName(args[0]);
            Method m[] = c.getDeclaredMethods();
            for (int i = 0; i < m.length; i++)
                System.out.println(m[i].toString());
        }
        catch (Throwable e) {
            System.err.println(e);
        }
    }
}
```

JAVA REFLECTION

ELŐNYÖK/HÁTRÁNYOK

- Előnyei, tipikus felhasználási területei
 - Bővíthetőség
 - Külső komponensek név alapján történő betöltése
 - Class browser és vizuális fejlesztői környezetek
 - Debugger és tesztelő eszközök
- Amennyiben valami reflection nélkül is megoldható, az a preferált!!!
- Reflection hátrányai
 - Teljesítmény csökkenés
 - Biztonsági kockázatok
 - Runtime permission
 - Belső adatok felfedése
 - Pl. private adattagok

ANNOTÁCIÓK

- Meta-adatok megadásának egy formája
- Olyan adatokat szolgáltat egy programhoz, ami alapvetően nem képezi magának a programnak a részét
- Nincs közvetlen hatásuk az annotált kód működésére
- A legegyszerűbb formájában egy annotáció így néz ki
 - `@Entity`
 - A `@` jelzi a fordító számára, hogy egy annotáció következik
 - Alkalmazható osztályokra, metódusokra, mezőkre
- Tartalmazhatnak egyéb elemeket
 - `@Author(name = "Jane Doe")`
- Egy deklarációt több annotációval is elláthatunk

```
@Author(name = "Jane Doe")
@EBook
class MyClass { ... }
```

ANNOTÁCIÓK FELHASZNÁLÁSA

- Annotációk tipikus felhasználása
 - Fordító számára információ megadása
 - Pl. hibák detektálásához, vagy figyelmeztetések kihagyásához
 - Fordítás idejű feldolgozás
 - Pl. kódgenerálás, XML fájl létrehozás, stb.
 - Futás idejű feldolgozás
 - Néhány annotáció futásidőben is elérhető
- Annotációk használhatók egy típus felhasználásakor is (nem csak a deklarációnál)
 - `new @Interned MyObject();`
 - `myString = (@NonNull String) str;`
 - `class UnmodifiableList<T> implements @ReadOnly List<@ReadOnly T> { ... }`
 - `void monitorTemperature() throws @Critical TemperatureException { ... }`

ELŐRE DEFINIÁLT ANNOTÁCIÓK

- Java nyelv által használt annotációk
 - **@Deprecated**
 - **@Override**
 - **@SuppressWarnings**
 - **@SafeVarargs**
 - **@FunctionalInterface**
- Annotációkra alkalmazható annotációk
 - **@Retention**
 - **@Documented**
 - **@Target**
 - **@Inherited**
 - **@Repeatable**

SAJÁT ANNOTÁCIÓ KÉSZÍTÉSE

- Deklaráció

```
import java.lang.annotation.Documented;
import java.lang.annotation.ElementType;
import java.lang.annotation.Inherited;
import java.lang.annotation.Retention;
import java.lang.annotation.RetentionPolicy;
import java.lang.annotation.Target;

@Documented
@Target(ElementType.METHOD)
@Inherited
@Retention(RetentionPolicy.RUNTIME)
public @interface MyCustomAnnotation{
    int studentAge() default 18;
    String studentName();
    String stuAddress();
    String stuStream() default "CSE";
}
```

- Felhasználás

```
@MyCustomAnnotation(
    studentName="Chaitanya",
    stuAddress="Agra, India"
)
public class MyClass {
    ...
}
```

RETENTION POLICY

- A **@Retention** annotációval lehet megadni
- Meghatározza, hogy az annotációk mikor érhetők el
 - RetentionPolicy.SOURCE
 - Csak forráskód szintjén érhetők el, nincs nyomuk a lefordított byte-kódban
 - RetentionPolicy.CLASS
 - Belekerül a byte-kódba, de a class betöltése után elveszik, azaz futási időben nincsen jelen az annotáció
 - Alapértelmezett érték
 - RetentionPolicy.RUNTIME
 - Futásidőben is elérhető annotáció (reflection-nel lekérdezhető)

SAJÁT ANNOTÁCIÓ FELDOLGOZÁSA

- Class objektum és reflection segítségével

```
1  Class aClass = MyClass.class;
2  Annotation[] annotations = aClass.getAnnotations();
3
4  for(Annotation annotation : annotations){
5      if(annotation instanceof MyCustomAnnotation){
6          MyCustomAnnotation myAnnotation = (MyCustomAnnotation) annotation;
7          System.out.println("name: " + myAnnotation.studentName());
8          System.out.println("address: " + myAnnotation.stuAddress());
9      }
10 }
```

JAVA GENERICS

- Generics = típusok használata paraméterként osztályok, interfészek vagy metódusok deklarálásakor
- Kód újra felhasználást segítik
- Hagyományos paraméterekkel ellentétben a típus paraméterek értékei típusok lesznek
 - A normál paraméterek értékei konkrét értékek
- Növelik a fordító által felfedhető hibák arányát
 - Erősebb ellenőrzés fordítás során
 - Cast műveletek kiiktatása
- Általános, típusfüggetlen algoritmusok írhatók

GENERIKUS OSZTÁLY, METÓDUS

- Nem generikus példa

```
public class Box {  
    private Object object;  
  
    public void set(Object object) { this.object = object; }  
    public Object get() { return object; }  
}
```

- Generikus megvalósítás

```
/**  
 * Generic version of the Box class.  
 * @param <T> the type of the value being boxed  
 */  
public class Box<T> {  
    // T stands for "Type"  
    private T t;  
  
    public void set(T t) { this.t = t; }  
    public T get() { return t; }  
}
```

GENERIKUS OSZTÁLY, METÓDUS HÍVÁSA

- Generikus osztály használata

- vagy

```
Box<Integer> integerBox = new Box<Integer>();
```

```
Box<Integer> integerBox = new Box<>();
```

- Generikus metódusok

```
Pair<Integer, String> p1 = new Pair<>(1, "apple");  
Pair<Integer, String> p2 = new Pair<>(2, "pear");  
boolean same = Util.<Integer, String>compare(p1, p2);
```

- vagy akár

```
Pair<Integer, String> p1 = new Pair<>(1, "apple");  
Pair<Integer, String> p2 = new Pair<>(2, "pear");  
boolean same = Util.compare(p1, p2);
```

```
public class Util {  
    public static <K, V> boolean compare(Pair<K, V> p1, Pair<K, V> p2) {  
        return p1.getKey().equals(p2.getKey()) &&  
            p1.getValue().equals(p2.getValue());  
    }  
}
```

```
public class Pair<K, V> {  
  
    private K key;  
    private V value;  
  
    public Pair(K key, V value) {  
        this.key = key;  
        this.value = value;  
    }  
  
    public void setKey(K key) { this.key = key; }  
    public void setValue(V value) { this.value = value; }  
    public K getKey() { return key; }  
    public V getValue() { return value; }  
}
```

JAVA 7 ÉS 8 HASZNOS ÚJDONSÁGAI

- Java7
 - Try-with-resources
 - Try multiple catch
- Java 8
 - Lambda kifejezések
 - Collection API
 - Stream API
 - forEach
 - Default methods
 - Security
- Java 9
 - Modul rendszer (Jigsaw)
 - Linkelés
- Java 10 (11) 😊
 - var // type inference

TRY-WITH-RESOURCES

- Egy speciális try utasítás, ami erőforrásokat deklarál
 - Az erőforrás egy olyan objektum, amit a használat végeztével le kell zárni
 - A konstrukció biztosítja minden ilyen erőforrás lezárását az utasítás végén
 - Bármilyen, ami implementálja a `java.lang.AutoCloseable` interfészt, használható erőforrásként
 - Az erőforrás deklaráció a `try` kulcsszó után zárójelek közt jelenik meg
 - A lezárás normális és kivételes viselkedés esetén is megtörténik
 - Több erőforrást is definiálhatunk

```
static String readFirstLineFromFile(String path) throws IOException {  
    try (BufferedReader br =  
        new BufferedReader(new FileReader(path))) {  
        return br.readLine();  
    }  
}
```

TRY MULTIPLE CATCH

- Egy `catch` blokk több különböző hiba típust tud egyszerre kezelni
 - Kód duplikáció kiküszöbölése
 - Túl általános hibák elkapásának megelőzése

```
catch (IOException ex) {  
    logger.log(ex);  
    throw ex;  
    catch (SQLException ex) {  
        logger.log(ex);  
        throw ex;  
    }  
}
```



```
catch (IOException|SQLException ex) {  
    logger.log(ex);  
    throw ex;  
}
```

LAMBDA KIFEJEZÉSEK

- Név nélküli függvények
- Az egy metódust tartalmazó anonymous belső osztályok kiváltására
 - Functional interface
 - Olyan interfész, aminek egyetlen absztrakt metódusa van (pl. Runnable, ActionListener, stb.)
- Funkcionalitás mint metódus argumentum, azaz kód, mint adat
- Felépítésük
 - Argumentum lista -> törzs
 - `(int x, int y) -> x + y`
- Funkcionális programozási képességek Java-ban

FUNCTIONAL INTERFACE

- Interfész egyetlen absztrakt metódussal
- `@FunctionalInterface` annotáció
- Előre definiált interfészek
 - Function
 - Supplier
 - Consumer
 - Predicate
 - Operator

```
1 | Function<Integer, String> intToString = Object::toString;
2 | Function<String, String> quote = s -> "'" + s + "'";
3 |
4 | Function<Integer, String> quoteIntToString = quote.compose(intToString);
5 |
6 | assertEquals("'5'", quoteIntToString.apply(5));
```

LAMBDA KIFEJEZÉS PÉLDÁK

- Interfész megadása

```
6 public class RunnableTest {  
7     public static void main(String[] args) {  
8  
9         System.out.println("=== RunnableTest ===");  
10  
11         // Anonymous Runnable  
12         Runnable r1 = new Runnable() {  
13  
14             @Override  
15             public void run() {  
16                 System.out.println("Hello world one!");  
17             }  
18         };  
19  
20         // Lambda Runnable  
21         Runnable r2 = () -> System.out.println("Hello world two!");  
22  
23         // Run em!  
24         r1.run();  
25         r2.run();  
26  
27     }  
28 }
```

- Rendezéshez

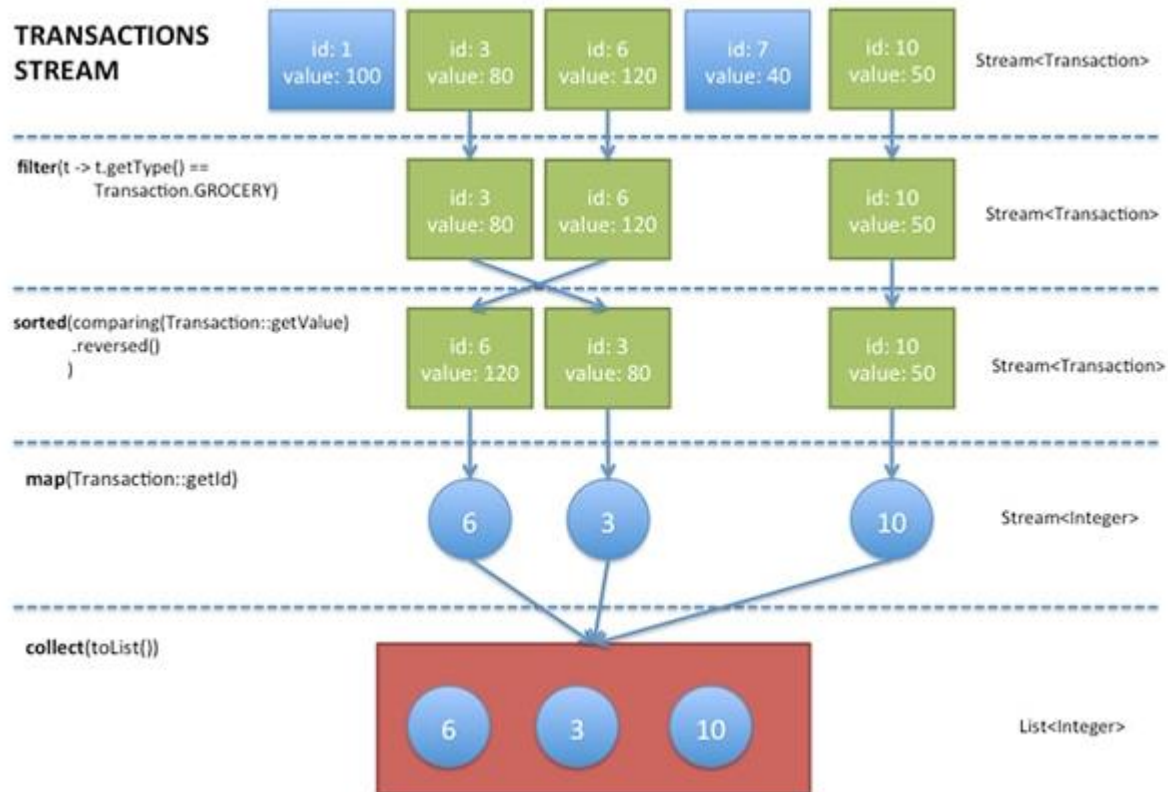
```
// Print Desc  
System.out.println("=== Sorted Desc SurName ===");  
Collections.sort(personList, (p1, p2) -> p2.getSurName().compareTo(p1.getSurName()));
```

STREAM API

- Kollektciók (soros) folyamammá (stream) alakítása
- Becsomagolják a kollektciókat, és plusz műveleteket adnak hozzájuk
 - A mit-re lehet fókuszálni a hogyan helyett
- Egy sor klasszikus feladat kompakt módon, deklaratívan megadható
 - Szűrés
 - Csoportosítás
 - Map-reduce
- Erősen épít a lambda kifejezésekre
- A legtöbb stream művelet stream-et ad vissza, így a műveletek láncolhatók
- Tipikus használat
 1. Stream létrehozása
 2. Néhány köztes művelet végrehajtása (újabb stream-ek jönnek létre)
 3. Befejező művelet végrehajtása (nem hoz létre újabb stream-et és/vagy mellékhatása van)

STREAM PÉLDA

```
List<Integer> transactionsIds =  
    transactions.stream()  
        .filter(t -> t.getType() == Transaction.GROCERY)  
        .sorted(comparing(Transaction::getValue).reversed())  
        .map(Transaction::getId)  
        .collect(toList());
```



FOREACH

- Kollektciók kompaktabb bejárásához
- `Consumer` funkcionális interfész paraméter
 - Azaz lambda kifejezéssel megadható

```
1 List<String> names = new ArrayList<>();
2 names.add("Larry");
3 names.add("Steve");
4 names.add("James");
5 names.add("Conan");
6 names.add("Ellen");
7
8 names.forEach(new Consumer<String>() {
9     public void accept(String name) {
10         System.out.println(name);
11     }
12 });
```

– vagy

```
1 names.forEach(name -> System.out.println(name));
```

– vagy

```
1 names.forEach(System.out::println);
```

COLLECTION API

- Map közvetlen rendezése
 - Stream-ek segítségével
- Iteráció `forEach` segítségével
- `getOrDefault`
 - Nincs szükség if-else null ellenőrzésre
- `putIfAbsent`
 - Map kulcsok felülírása kivédhető
- Értékek közvetlen feldolgozása
 - Nincs szükség az érték lekérésére, módosítására, visszarakására
- Merge map-hez

DEFAULT METHODS

- Interface metódusok tartalmazhatnak alapértelmezett törzset
- Lehetőséget ad új metódusok deklarálására meglévő interfészekben anélkül, hogy elrontaná a meglévő implementációt

```
@FunctionalInterface
public interface Iterable<T> {
    Iterator<T> iterator();

    default void forEach(Consumer<? super T> action) {
        Objects.requireNonNull(action);
        for (T t : this) {
            action.accept(t);
        }
    }
}
```

SECURITY

- TLS 1.1 és TLS 1.2 alapértelmezetten engedélyezett
- Korlátozott `doPrivileged`
- Erősebb jelszó titkosító algoritmusok
- KeyStore fejlesztések
- SHA-224
- Nagy entrópiájú véletlen szám generátor
- Megbízhatóbb tanúsítvány ellenőrzés
- Stb.

JAVA MODULOK

- Újabb absztrakciós szint
 - Csomag szintű függőségek
 - A jar fájlaktól finomabb granularitás
- Moduláris jar fájlok
- Requires/export
- Core osztályok

```
module kocsi {  
    exports hu.lamborghini.kocsi;  
  
    requires motor;  
}
```

JAVA 9 LINKELÉS

- Jlink
- Kompakt profilok
- Megtakarított tárhely

JAVA 10 LOKÁLIS VÁLTOZÓ TÍPUS KIKÖVETKEZTETÉS

- A var kulcsszó bevezetése típus automatikus kikövetkeztetéséhez
 - Fejlesztői élmény növelése
 - „Syntactic sugar”
- Változó deklarálása a típus explicit megnevezése nélkül

```
List<String> list = new ArrayList<String>();  
Stream<String> stream = getStream();
```



```
var list = new ArrayList<String>();  
var stream = getStream();
```

KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

FELHASZNÁLÓ ÉS JELSZÓADATOK BIZTONSÁGOS KEZELÉSE

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI  2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



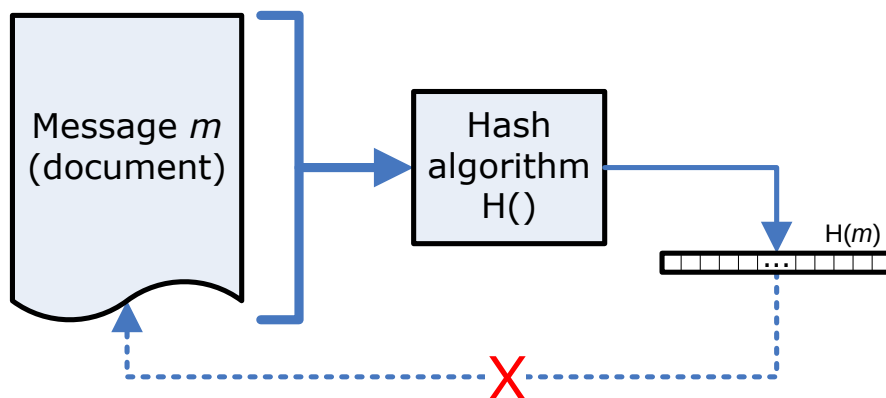
BEFEKTETÉS A JÖVŐBE

BEVEZETÉS

- A legtöbb információs rendszerben előforduló feladat
 - Felhasználók kezelése
 - Authentikáció megvalósítása
 - Felhasználói adatok tárolása (beleértve jelszó is)
 - Az egyes erőforrások hozzáféréseinek korlátozása, szabályozása
- Megoldandó problémák jellege
 - Biztonsági
 - Technikai
 - Jogi (adatvédelem)

HASH VAGY MESSAGE DIGEST

- Kriptográfiai hash függvény: egy determinisztikus eljárás, amely egy fix méretű bit string-et állít elő adatok blokkjából



- A kriptográfiai hash függvények fő tulajdonságai
 - A hash érték könnyen számítható
 - Nehéz egy konkrét hash értékkel bíró üzenetet találni
 - Nehéz két olyan üzenetet találni, amelyekhez ugyanaz a hash tartozik
 - Az üzenetben akár már 1 bit megváltoztatása is jelentősen módosítja az előálló hash értéket (avalanche effektus)

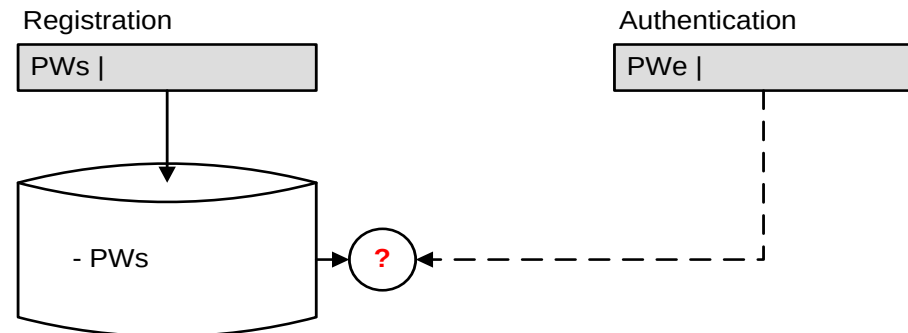
HASH ALGORITMUSOK

- **MD5** – Message Digest 5 (16 bytes, 128 bits)
FELTÖRT – gyengének tekinthető
- **SHA-1** – Secure Hash Algorithm (20 bytes, 160 bits)
Egyelőre ≈OK az SSL/TLS handshake-hez, ill. ha NEM használják digitális aláíráshoz
- **SHA-2** – Algoritmus család (különböző variánsok, 2001-)
 - SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256 – *Mind OK, SHA-256 jó választás*
- **SHA-3** – A legújabb NIST kihívás győztese (2015-)
 - SHA3-224, SHA3-256, SHA3-384, SHA3-512 – *Mind OK*
- **Argon2, PBKDF2, bcrypt** vagy **scrypt** mindegyike kifejezetten jelszavak tárolására használatos

JELSZÓ KEZELÉS – HOL A HIBA?

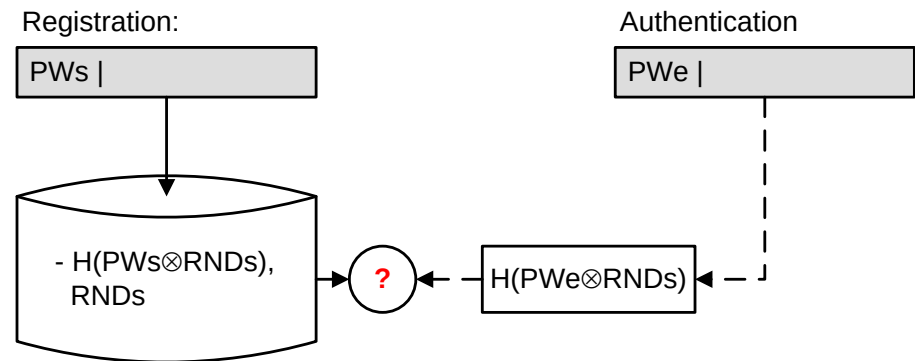
```
• PreparedStatement ps=conn.prepareStatement("SELECT * FROM users WHERE username=? AND  
  userpass=?");  
ps.setString(1, request.getParameter("username"));  
ps.setString(2, request.getParameter("password"));  
ResultSet rst = ps.executeQuery();  
if (rst.next())  
{  
    session.setAttribute("userid", rst.getInt(5));  
}  
else  
{  
    out.print("Invalid credentials. Please try again.<br/>");  
}
```

- Naiv megközelítés: jelszavak tárolása
sima szöveggént
- Mi történik, ha a jelszó adatbázist
ellopják?
 - A támadó minden felhasználói
adathoz egyetlen behatolással
hozzájut
- Megoldás: hash-elés



JELSZÓ KEZELÉS ÉS TÁROLÁS

- A hash-elt jelszavak még mindig sérülékenyek
 - A jelszó találgatásra: üres, felhasználó név, stb.
 - Szótár alapú támadásokra
 - Googling-gel támogatva
 - Brute force támadásokra
 - Előre kiszámított Rainbow táblákkal támogatva
- Megelőzés
 - Jelszó házirend
 - Lassú hash algoritmusok használata, mint pl. a bcrypt vagy PBKDF2
 - Salting:



JELSZÓ TÁROLÁSRA ALKALMAZHATÓ SPECIÁLIS HASH ALGORITMUSOK

- **Argon2**
 - A 2015-ös Password Hashing Competition győztese
 - OWASP ajánlás szerint a #1 választás (használjuk ezt, ha elérhető)
- **Password-Based Key Derivation Func. 2 (PBKDF2)**
 - Az SHA1 vagy SHA256 (HMAC) algoritmuson alapul, az iterációk szabályozhatók
 - A NIST által ajánlott, FIPS-140 megfelelés
- **bcrypt**
 - A Blowfish titkosító algoritmuson alapszik, salting alapból benne van
 - A sebessége szabályozható
 - GPU támadásokkal szemben ellenállóbb (globális táblát használ)
- **scrypt**
 - Hosszú string-eket manipulál a memóriában
 - Robusztusabb és lassabb, mint a PBKDF2 vagy bcrypt

ARGON2 ÉS PBKDF2 IMPLEMENTÁCIÓK JAVA-BAN

- **Argon2**
 - Argon2-JVM
 - A wrapper for the C library
 - <https://github.com/phxql/argon2-jvm>
- **PBKDF2**
 - JCE (`javax.crypto`)
 - `SecretKeyFactory` should be instantiated with the "PBKDF2WithHmacSHA1" algorithm name
 - Implemented in BouncyCastle
 - `PBKDF2WithSHA1*`, multiple classes supporting UTF8 and ASCII encoding

BCRYPT ÉS SCRIPT JAVA MEGVALÓSÍTÁSOK

- **bcrypt**
 - A BouncyCastle provider implementálja (BCrypt osztály)
 - Spring-ben
 - `org.springframework.security.crypto.bcrypt`
 - BCrypt class
 - BCryptPasswordEncoder osztály
 - JBCrypt (3rd party)
- **script**
 - BouncyCastle
 - Lambdaworks SCrypt, 3rd party könyvtár
 - <https://github.com/wg/script>

Esettanulmány – az Ashley Madison
adatlopás

ESETTANULMÁNY – AZ ASHLEY MADISON ADATLOPÁS

- 2015 augusztusában az Ashley Madison „randioldalt” feltörték, és lopott adatok kerültek nyilvánosságra
 - Tagok adatai a tranzakciókat beleértve (kártyaadatok nem)
 - 36 millió felhasználói jelszó
 - Bcrypt algoritmussal védve, 12-es költséggel ($2^{12}=4096$ iteráció)
- Kutatók brute force támadást intéztek a bcrypt ellen
 - Szótáras támadás 4 GPU-val
 - A leggyengébb **4007** jelszót sikerült feltörniük **5 nap alatt**
 - Mind a 36M jelszó feltörése csak szótáras módszerrel több mint egy évszázadba telt volna!
- Azonban a rendszer forráskódja is kiszivárgott
 - Felfedve egy használható brute force módszert a jelszavak feltörésére

A LOGINKEY TOKEN

- A loginkey MD5 algoritmussal generálódott, amit az adatbázisban is **tároltak**
 - Valószínűleg automatikus beléptetéshez használták
- Ez a kulcs regisztrációkor ill. profil módosításkor generálódott
 - A korai időszakban így:

```
md5(lc($username).":":lc($pass))
```

- Később a generálást erre cserélték:

```
md5(lc($username).":":lc($pass).":":lc($email).":73@^bhhs&#@&^@8@*$")
```

- Végül, 2012-ben egy sokkal jobb megoldást választottak:

```
md5(lc($username).":":lc(bcrypt($pass)))
```

- Azonban ezt csak az újonnan regisztráltaknál alkalmazták
 - A már meglévő felhasználók kulcsait meghagyták a régi, sérülékeny formában

JELSZAVAK FELTÖRÉSE BRUTE FORCE MÓDSZERREL

- Brute force módszerrel támadhatjuk az MD5 hash-t (loginkey) a bcrypt helyett
 - A sima szöveg jelszó az egyetlen ismeretlen az MD5 hash-ben
 - Mivel minden kisbetűs, még hatékonyabb is
- Kis/nagybetű javítás szükséges a végén
 - A feltört (lehetséges) jelszó csupa kisbetűs
 - Minden karakter egyesével kicserélhető nagyra (a bcrypt értékkel összevethetők kombinációk), hogy megtaláljuk a valós jelszót
- A 2012 előtt regisztrált felhasználók jelszavai közül **11,2 millió** bcrypt-el verziót sikerült feltörni néhány nap alatt
- Leckék
 - Soha ne használjuk a sima szöveges jelszót token hash-ekben
 - Ha cseréljük a jelszó hash-elést erősebbre, visszamenőleg minden értéket migráljunk

JELSZÓ TÁROLÁS TIPIKUS HIBÁI

- Gyenge titkosítás
 - A gyengén titkosított jelszavak nem védettek
- Jelszó levágás
 - A hosszú jelszavak biztonságának csorbítása
- Jelszó naplózása
 - Ne naplózzunk semmilyen jelszót, még a helyteleneket sem
- Alapértelmezett jelszavak
 - Ne használjuk ugyanazt a jelszót mindenütt, vagy legalább vegyük rá a felhasználókat, hogy időnként lecseréljék
- Beégetett jelszavak
 - A forráskódban elhelyezett jelszavak könnyen felfedhetők
 - **A forráskódra mindig „nyilvános adatként” tekintünk**

KLIENS OLDALI AUTHENTIKÁCIÓ ÉS JELSZÓ KEZELÉS

- Tipikus hibák
 - Kliens oldali autentikáció vagy biztonsági funkcionalitás
 - Beégetett jelszavak
- Jó praktikák
 - Mindig szerver oldali autentikációt használjunk
 - Ne használjunk beégetett jelszavakat a JavaScript kódban, még titkosítva sem
 - A hackerek mindig megtalálják a módját, hogy ezeket kiderítsék

KLIENS OLDALI BIZTONSÁG – HOL A HIBA?

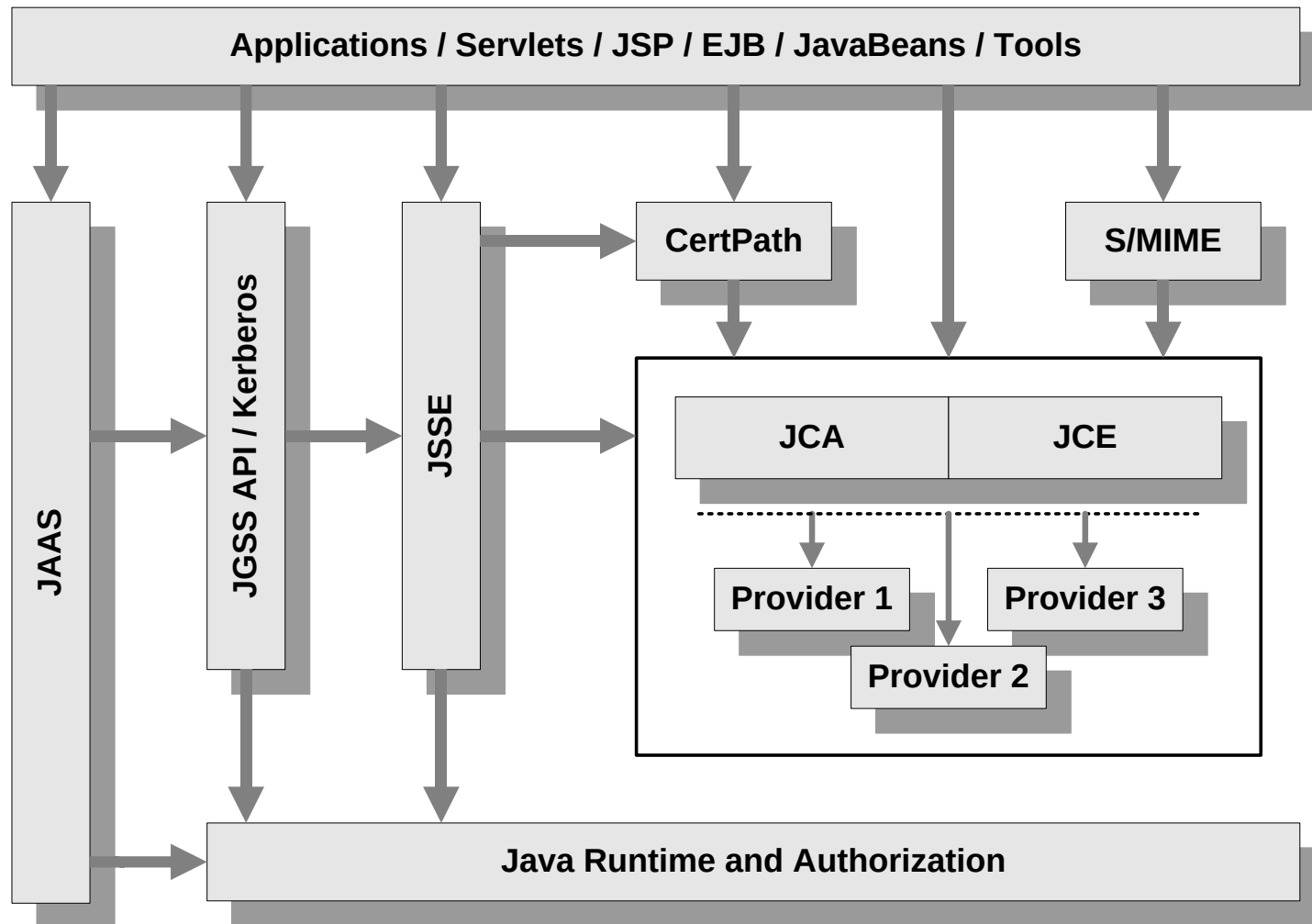
```
function check()
{
    if ( SHA1(document.auth.pass.value) ==
        '43206512b209ba29cb5c642edc85bdac133354fe' )
    {
        window.location.href="admin_page.jsp";
    }
    else
    {
        alert('Invalid password!');
    }
}
```

- **Kliens oldali biztonság nem biztonság!**
 - Ne támaszkodjunk semmi féle kliens oldali ellenőrzésre/biztonsági funkcionálisra
 - A bemenet ellenőrzése kliens oldalon csökkentheti a hálózati forgalmat
 - De meg kell ismételni szerver oldalon is!

JAVA BIZTONSÁGI SZOLGÁLTATÁSOK

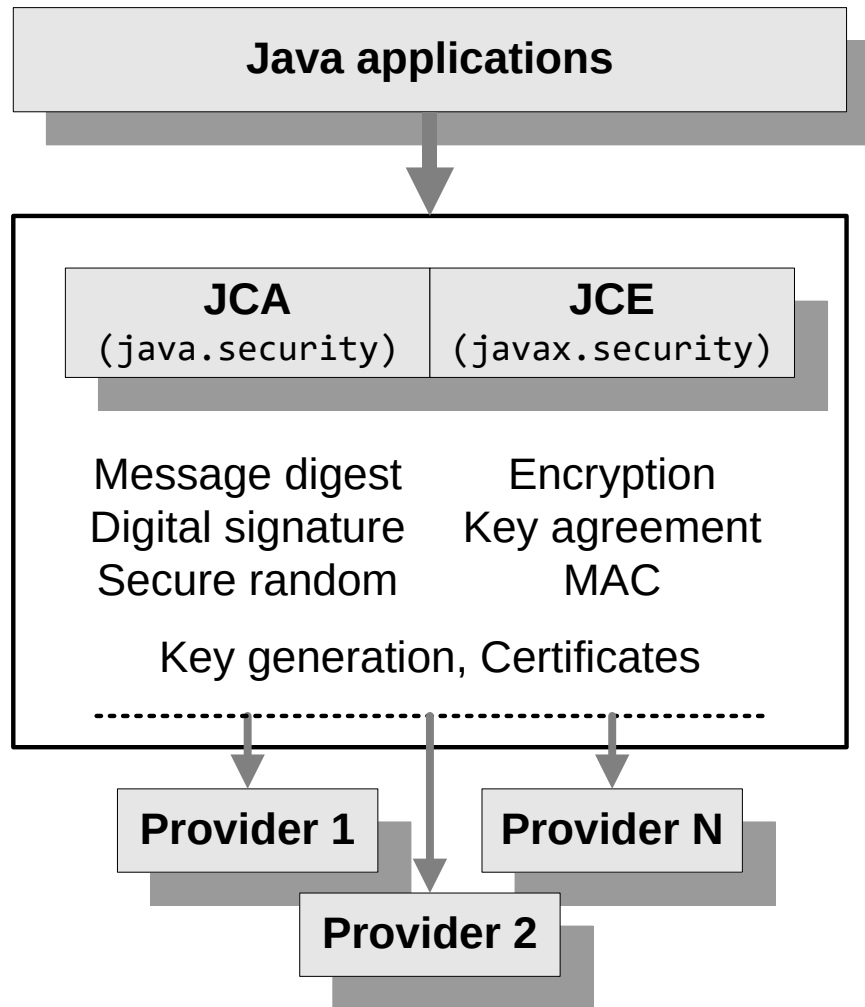
- Független az implementációtól
 - Az alkalmazások nem implementálnak biztonsági algoritmusokat
 - A biztonsági szolgáltatásokat ún. provider-ek biztosítják
- Felcserélhetőség
 - Egy alkalmazás nincs egyetlen provider-hez sem kötve
 - Egy provider sincs egyik alkalmazáshoz sem kötve
- Algoritmusok kiterjeszthetősége
 - A Java platform számos beépített provider-t tartalmaz
 - De támogatja saját provider-ek telepítését is

JAVA BIZTONSÁGI SZOLGÁLTATÁSOK – ARCHITEKTÚRA



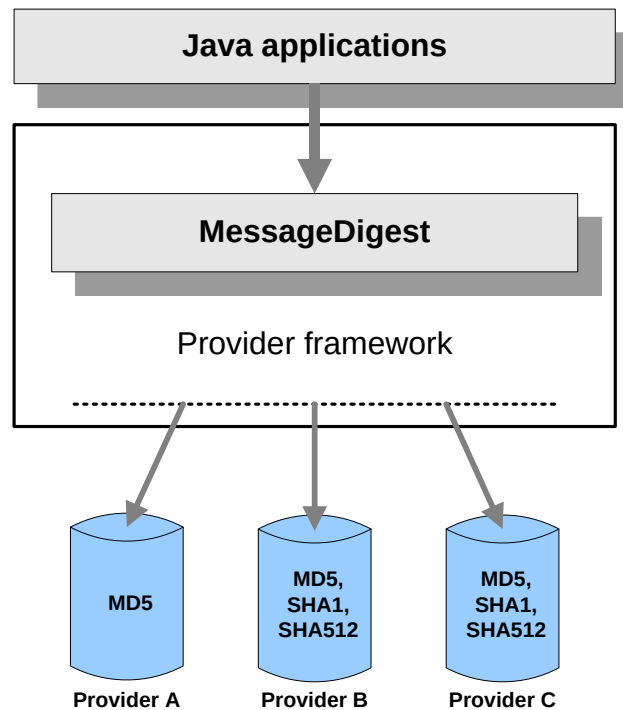
Java Cryptographic Architecture

JAVA CRYPTOGRAPHY ARCHITECTURE / EXTENSION (JCA/JCE)



KRIPTOGRÁFIAI PROVIDER-EK HASZNÁLATA

```
hash = MessageDigest.getInstance("SHA1", "ProviderC");
```



FŐ OSZTÁLYOK ÉS ALGORITMUSOK

- SecureRandom
 - Kriptográfiailag erős véletlen számok generálása
- MessageDigest
 - Egy adat hash értékének kiszámítása
- Signature
 - Privát és publikus kulcsokkal inicializáljuk, adatok aláírására és aláírások ellenőrzésére használható
- Cipher
 - Kulcsokkal inicializáljuk, titkosításra/visszafejtésre használható
 - AES, DES, DESede, Blowfish, IDEA, RC4, RSA, PBE
- MAC (Message Authentication Codes)
 - Kulcsokkal inicializálva hash értékek generálására szolgál (adat+kulcs)

FŐ OSZTÁLYOK ÉS ALGORITMUSOK

- **KeyPairGenerator**
 - Új publikus/privát kulcspár generálása specifikus algoritmusokhoz
- **KeyFactory**
 - Titkosító kulcsok Key típusának átalakítása kulcs specifikációvá
- **KeyGenerator**
 - Új privát kulcs generálása specifikus algoritmusokhoz
- **SecretKeyFactory**
 - Titkosító kulcsok SecretKey típusának átalakítása kulcs specifikációvá
 - (Szimmetrikus) titkosító kulcsokat gyárt

A BOUNCY CASTLE (BC) PROVIDER

- Szimmetrikus blokk titkosító algoritmusok
 - AES, Rijndael, Blowfish, IDEA, RC6, Serpent, Twofish, TEA, ...
- Szimmetrikus stream titkosító algoritmusok
 - RC4, HC256, Salsa20, ISAAC, VMPC, ...
- Aszimmetrikus algoritmusok
 - RSA, ElGamal
- Key Agreement algoritmusok
 - Diffie-Hellman (using DH, ECDH, ECCDH, ECMQV)
- Digest algoritmusok
 - MD2, MD4, MD5, GOST3411, RipeMD..., Tiger, Whirlpool, SHA1, SHA224, SHA256, SHA384, SHA512, SHA3, ...
 - Jelszó hash-elés: BCrypt, OpenBSDBcrypt, SCrypt
- MAC algoritmusok
 - VMPC-MAC, HMAC

SECURITY FRAMEWORKS

- Komplex keretrendszerek léteznek a fent vázolt problémák megoldására
 - Felhasználó azonosítás
 - Jelszó kódolás/tárolás
 - Hozzáférés vezérlés
- Java példák
 - Apache Shiro
 - Authentikáció több forrásból (LDAP, JDBC, ...)
 - Kriptográfiai algoritmusok
 - Session kezelés
 - Spring Security
 - Authentikáció, hozzáférés vezérlés
 - Sérülékenységek elleni védelem

KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

NOSQL ADATBÁZISOK

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

NOSQL

- Not only SQL/nem SQL/egyik sem? ☺
- Adatbázisok, melyek
 - Elsősorban nem táblákban tárolják az adatokat
 - Általában nem használnak SQL nyelvet lekérdezésre
 - Erősen optimalizáltak írás és olvasás műveletekre
 - Könnyen skálázhatók
 - Tárolt eljárások helyett a map-reduce programokat lehet rajtuk futtatni
- Több típusa is létezik a NoSQL adatbázisoknak

NOSQL

- Tipikusan nem támogatják az ACID tranzakciókat
 - Atomicity
 - Consistency
 - Isolation
 - Durability
- NoSQL gyakran feláldozza a konzisztenciát a rendelkezésreállásért (availability)
 - „Eventual consistency”
 - Vannak kivételek
- Könnyebb a horizontális skálázás
 - Adatok több gépen való szétterítése

KULCS-ÉRTÉK TÁROLÓK

- Séma nélküli adattárolás
- Az értékeket a kulcsok segítségével lehet elérni illetve felülírni
 - Ezen felül általában kevés másik műveletet támogatnak
 - Változó konzisztencia modell
 - Kulcsok rendezése, tartományok kezelése
- Implementációk
 - Riak
 - Cassandra
 - Redis
 - https://en.wikipedia.org/wiki/Key-value_database

REDIS

- Kulcs-érték adattár
 - Hashmap
- In-memory
- Elosztott
- Adatbázis, cache, message broker
- Bináris protokoll

PARANCSOK

APPEND
AUTH
BGREWRITEAOF
BGSAVE
BITCOUNT
BITFIELD
BITOP
BITPOS
BLPOP
BRPOP
BRPOPLPUSH
CLIENT KILL
CLIENT LIST
CLIENT GETNAME
CLIENT PAUSE
CLIENT REPLY
CLIENT SETNAME
CLUSTER ADDSLOTS
CLUSTER COUNT-FAILURE-REPORTS
CLUSTER COUNTKEYSINSLOT
CLUSTER DELSLOTS
CLUSTER FAILOVER
CLUSTER FORGET
CLUSTER GETKEYSINSLOT
CLUSTER INFO
CLUSTER KEYSLOT
CLUSTER MEET
CLUSTER NODES
CLUSTER REPLICATE
CLUSTER RESET
CLUSTER SAVECONFIG
CLUSTER SET-CONFIG-EPOCH
CLUSTER SETSLOT

CLUSTER SLAVES
CLUSTER SLOTS
COMMAND
COMMAND COUNT
COMMAND GETKEYS
COMMAND INFO
CONFIG GET
CONFIG REWRITE
CONFIG SET
CONFIG RESETSTAT
DBSIZE
DEBUG OBJECT
DEBUG SEGFAULT
DECR
DECRBY
DEL
DISCARD
DUMP
ECHO
EVAL
EVALSHA
EXEC
EXISTS
EXPIRE
EXPIREAT
FLUSHALL
FLUSHDB
GEOADD
GEOHASH
GEOPOS
GEODIST
GEORADIUS
GEORADIUSBYMEMBER

GET
GETBIT
GETRANGE
GETSET
HDEL
HEXISTS
HGET
HGETALL
HINCRBY
HINCRBYFLOAT
HKEYS
HLEN
HMGET
HMSET
HSET
HSETNX
HSTRLEN
HVALS
INCR
INCRBY
INCRBYFLOAT
INFO
KEYS
LASTSAVE
LINDEX
LINSERT
LLEN
LPOP
LPUSH
LPUSHX
LRANGE
LREM
LSET

LSET
LTRIM
MGET
MIGRATE
MONITOR
MOVE
MSET
MSETNX
MULTI
OBJECT
PERSIST
PEXPIRE
PEXPIREAT
PFADD
PFCOUNT
PFMERGE
PING
PSETEX
PSUBSCRIBE
PUBSUB
PTTL
PUBLISH
PUNSUBSCRIBE
QUIT
RANDOMKEY
READONLY
READWRITE
RENAME
RENAMEX
RESTORE
ROLE
RPOP
RPOPLPUSH

RPUSH
RPUSHX
SADD
SAVE
SCARD
SCRIPT DEBUG
SCRIPT EXISTS
SCRIPT FLUSH
SCRIPT KILL
SCRIPT LOAD
SDIFF
SDIFFSTORE
SELECT
SET
SETBIT
SETEX
SETNX
SETRANGE
SHUTDOWN
SINTER
SINTERSTORE
SISMEMBER
SLAVEOF
SLOWLOG
SMEMBERS
SMOVE
SORT
SPOP
SRANDMEMBER
SREM
STRLEN
SUBSCRIBE
SUNION

SUNIONSTORE
SWAPDB
SYNC
TIME
TOUCH
TTL
TYPE
UNSUBSCRIBE
UNLINK
UNWATCH
WAIT
WATCH
ZADD
ZCARD
ZCOUNT
ZINCRBY
ZINTERSTORE
ZLEXCOUNT
ZRANGE
ZRANGEBYLEX
ZREVRANGEBYLEX
ZRANGEBYSCORE
ZRANK
ZREM
ZREMRANGEBYLEX
ZREMRANGEBYRANK
ZREMRANGEBYSCORE
ZREVRANGE
ZREVRANGEBYSC
ZREVRANK
ZSCORE
ZUNIONSTORE

SCAN
SSCAN
HSCAN
ZSCAN

WEBDIS

- “A fast HTTP interface for Redis”
 - REST wrapper Redis fölé
- Egyetlen paraméterezett url
 - `/COMMAND/arg0/arg1/.../argN[.ext]`
 - ext: *json/txt/xml/msg/png/...*
- Publish/Subscribe támogatás
 - HTML5 Websocket
- ACL
 - webdis.json

PÉLDÁK

```
$ curl http://127.0.0.1:7379/GET/y  
{ "GET": "41" }
```

```
$ curl http://127.0.0.1:7379/INCR/y  
{ "INCR": 42 }
```

```
$ curl http://127.0.0.1:7379/SET/false/true  
{ "SET": [true, "OK"] }
```

```
$ curl http://127.0.0.1:7379/SET/true/false  
{ "SET": [true, "OK"] }
```

KLIENSEK

- Bash
 - `redi.sh`
- Python
 - Pottery
- Matlab
 - `go-redis`
- Oracle
 - OREDIS
- Redis

PYTHON – POTTERY PÉLDA

```
from pottery import RedisDict
redis = Redis.from_url('http://localhost:6379/')
spells = RedisDict(redis=redis, key='spells')

spells['wingardium'] = 'leviosaaah'
spells['staaahp'] = False

len(spells)           # 2
spells['staaahp']     # False
```



DOKUMENTUMTÁROLÓK

- Központi koncepció a „dokumentum”
 - A tárolandó adat egy egységét foglalja magában
- Tipikus formátumok
 - XML
 - YAML
 - JSON
 - BSON
- Az egyes tárolt dokumentumok nem kell, hogy ugyanolyan struktúrájúak legyenek
 - Nem feltétlenül ugyanazok a mezők, mint pl. egy relációs adatbázis rekordjainál

DOKUMENTUMTÁROLÓK

- A dokumentumok egyedi kulcsokkal vannak azonosítva
- A keresés viszont nem csak kulcs alapján történhet
 - Lekérdező nyelv + API a tartalom alapján történő keresésre
- Különböző implementációk különböző módon csoportosítják/szervezik a dokumentumokat
 - Gyűjtemények
 - Címkék
 - Meta-információk
 - Hierarchikus elrendezés

DOKUMENTUMTÁROLÓK

- Példa dokumentumok

```
{  
  "FirstName": "Bob",  
  "Address": "5 Oak St.",  
  "Hobby": "sailing"  
}
```

```
<contact>  
  <firstname>Bob</firstname>  
  <lastname>Smith</lastname>  
  <phone type="Cell">(123) 555-0178</phone>  
  <phone type="Work">(890) 555-0133</phone>  
  <address>  
    <type>Home</type>  
    <street1>123 Back St.</street1>  
    <city>Boys</city>  
    <state>AR</state>  
    <zip>32225</zip>  
    <country>US</country>  
  </address>  
</contact>
```

- Implementációk

- CouchDB
- Elasticsearch
- MongoDB
- https://en.wikipedia.org/wiki/Document-oriented_database

GRÁF ADATBÁZISOK

- Adatok tárolása, amelyek jól modellezhetők gráfként
 - Az adatelemek határozatlan számú kapcsolattal vannak összekötve
- Példák
 - Közösségi háló
 - Közlekedési térkép
 - Hálózati topológiák
 - Wiki oldalak
- Implementációk
 - Neo4j
 - OrientDB
 - MarkLogic
 - https://en.wikipedia.org/wiki/Graph_database

OBJEKTUM-ADATBÁZISOK

- Az objektum-orientált programozásban megszokott objektumokat tárolnak
 - OODBMS
- Különböznek a relációs adatbázisoktól
 - Azok tábla-orientáltak
 - Az objektum-relációs adatbázis a kettő ötvözete
- Általában nyújtanak valamilyen lekérdező nyelvet is
- Implementációk
 - JADE
 - ObjectDB
 - Stb.

EGYÉB NOSQL ADATBÁZIS TÍPUSOK

- Oszlop-orientált (column)
 - Apache Hbase
- Tuple store
 - Apache River
- Hosted megoldások
 - Amazon DynamoDB
 - CouchDB
 - Microsoft Azure DocumentDB
- Multimodel adatbázisok
 - Couchbase
 - OrientDB

NOSQL TELJESÍTMÉNY

- A különböző NoSQL típusok teljesítményének összehasonlítása

Data model ↕	Performance ↕	Scalability ↕	Flexibility ↕	Complexity ↕	Functionality ↕
Key–value store	high	high	high	none	variable (none)
Column-oriented store	high	high	moderate	low	minimal
Document-oriented store	high	variable (high)	high	low	variable (low)
Graph database	variable	variable	high	high	graph theory
Relational database	variable	variable	low	moderate	relational algebra

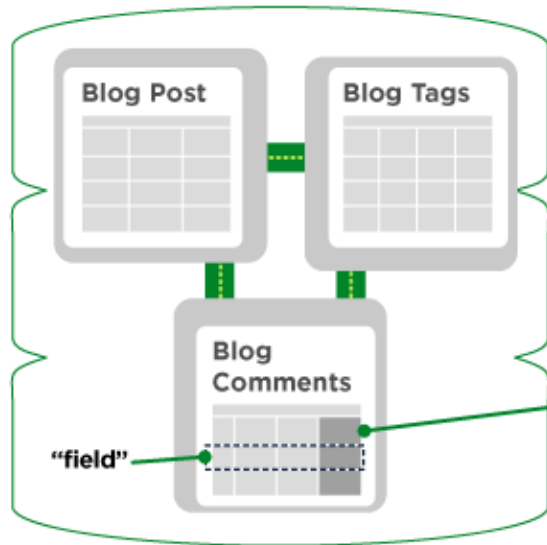
<https://en.wikipedia.org/wiki/NoSQL>

NOSQL VS. RDBMS

- Adatmodell
 - NoSQL esetén nincs (nem feltétlenül kell) előre definiált séma
- Adat struktúra
 - RDMS jól definiált tábla struktúrát és kapcsolatokat használ, NoSQL képes nem strukturált adatok (szöveg, email, video) tárolására
- Skálázhatóság
 - NoSQL esetén könnyebb a skálázás, eleve elosztott módon működnek, RDMS esetén cluster + erős szerver kell

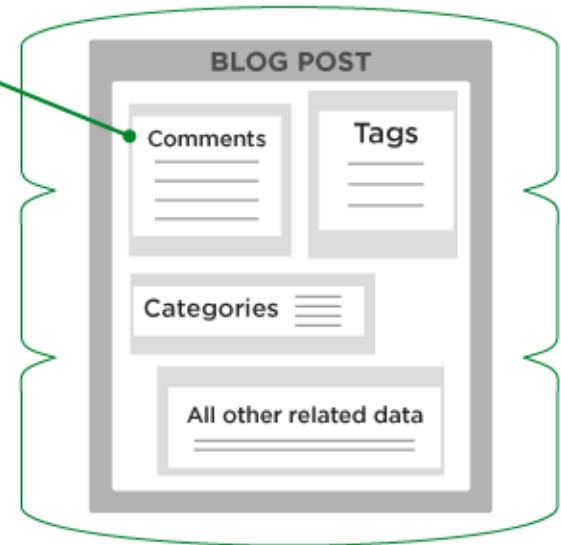
NOSQL VS. RDBMS

RELATIONAL VS. NON-RELATIONAL DATABASES



A non-relational database does not incorporate the table model. Instead, data can be stored in a single document file.

A relational database table organizes structured data fields into defined columns.



NoSQL-hez kapcsolódó fogalmak

MAP-REDUCE

Fault-tolerance

by @jrecursive



MAP-REDUCE

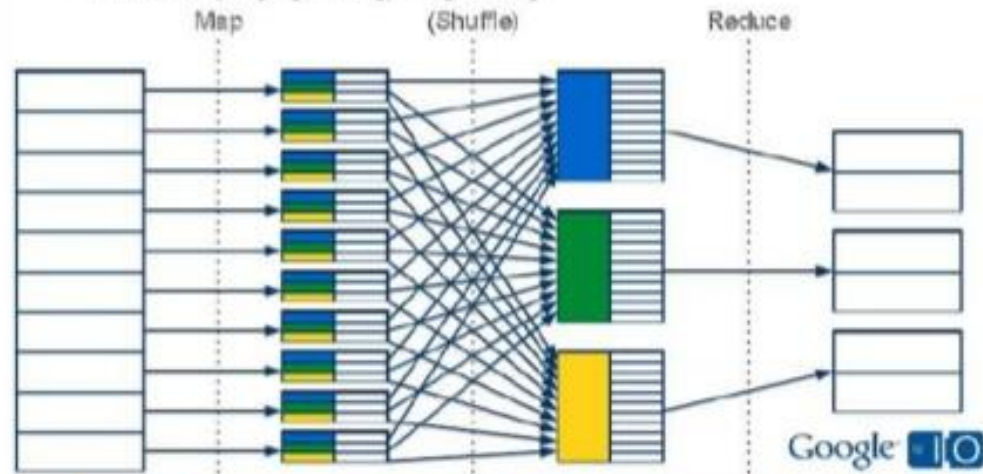
- Egy programozási modell
 - Funkcionális nyelvekből ered
 - Java8 is támogatja a Stream API segítségével
- Könnyű elosztott környezetben is futtatni
- A programozó két függvényt implementál
 - Map
 - Reduce
- Tetszőleges támogatott nyelven
 - Vagy egy keretrendszer segítségével
 - Skálázható/hibatűrő
 - Pl. Hadoop MapReduce

MAP-REDUCE FLOW

MapReduce Flow

MapReduce

- Developer defines only 2 functions:
 - `map(entity) -> [(key, value)]`
 - `reduce(key, [value]) -> [value]`



MAP-REDUCE PÉLDA

The prototypical MapReduce example counts the appearance of each word in a set of documents:

```
function map(String name, String document):  
    // name: document name  
    // document: document contents  
    for each word w in document:  
        emit (w, 1)  
  
function reduce(String word, Iterator partialCounts):  
    // word: a word  
    // partialCounts: a list of aggregated partial counts  
    sum = 0  
    for each pc in partialCounts:  
        sum += pc  
    emit (word, sum)
```

MAP-REDUCE PÉLDA

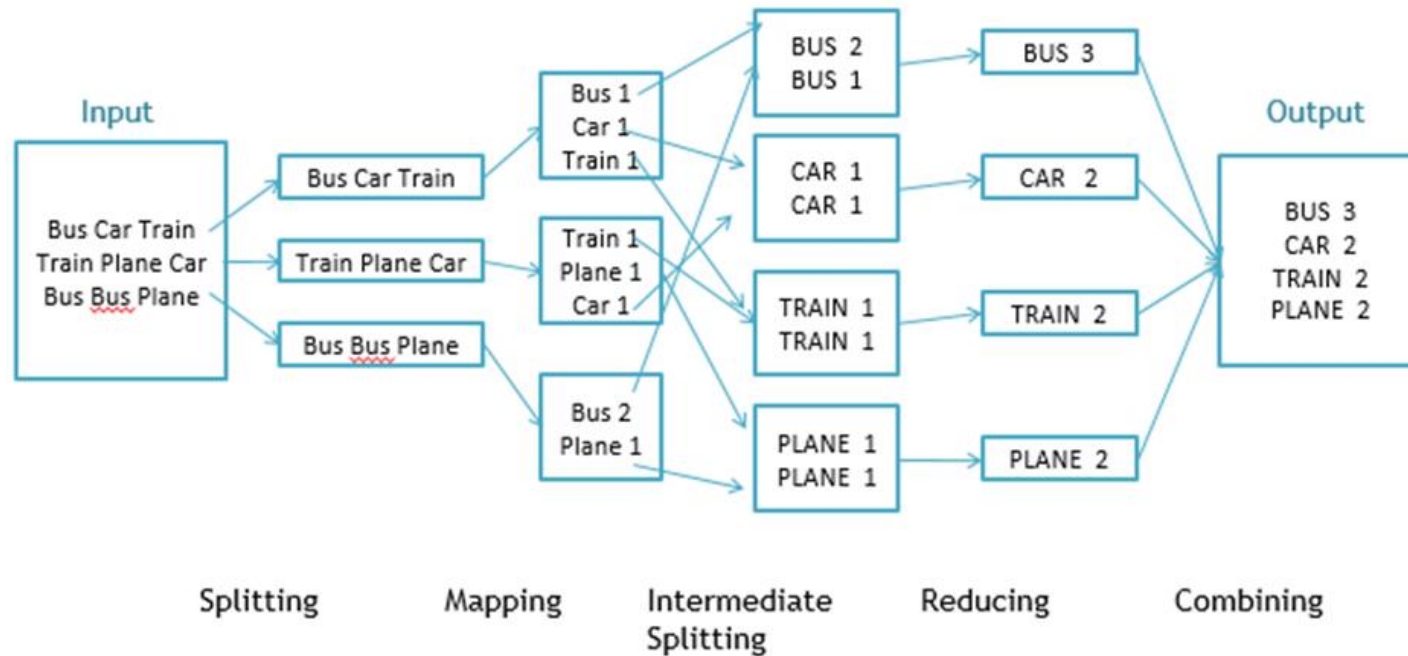


Fig. WorkFlow of MapReducing

ELOSZTOTT CACHE

- Performancia növelését célozza
- Gyakran használt adatok memóriában tartása
- Elosztott cache
 - A tradicionális lokális cache kiterjesztése
 - Több gépen átívelhet, skálázható
- Sok hasonlóság a NoSQL adatbázisokkal
- Némely NoSQL implementáció cache-ként is használható
- Néhány elosztott cache implementáció
 - Memcached
 - Redis (in-memory NoSQL)

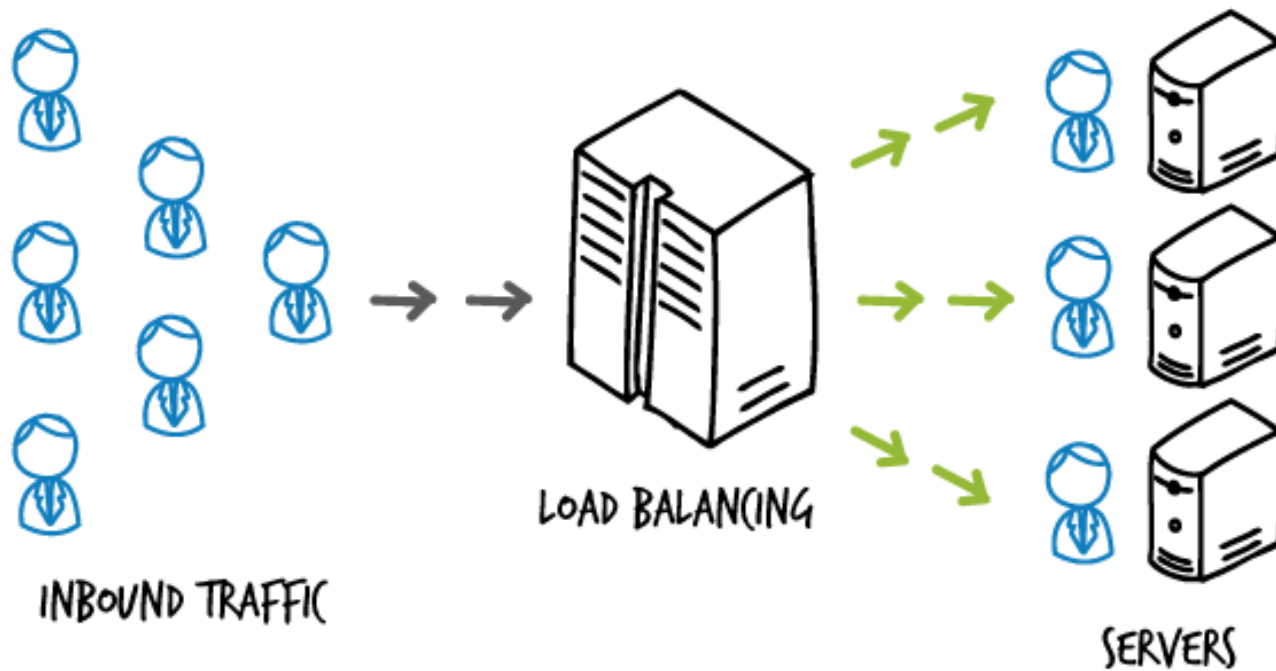
SHARDING

- Az adatok horizontális partícionálása
 - Minden partíció egy „shard”
- A megszokott vertikális partícionálással szemben itt az adat sorokat partícionáljuk
- A különböző adatsorok fizikailag is más-más szerveren lehetnek
- Sharding támogatás
 - Apache Hbase
 - MongoDB
 - Couchbase
 - RDMS-ek is támogathatják

SHARDING KIHÍVÁSOK

- Hálózati kapcsolat szerepe hangsúlyosabb
- Lekérések késleltetése megnőhet
- Az adatindexek gyakran csak egy irányba működnek
 - Bizonyos lekérdezések nagyon gyorsak
 - Mások nagyon lassúak/lehetetlenek
- Consistency és durability problémák
 - Bonyolult hibamódok miatt
- Elosztott környezetben hasznos lehet
- Komplex megvalósítani

LOAD BALANCING



<https://www.ctl.io/assets/images/blog/load-balancing.png>

KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

VÁLLALATIRÁNYÍTÁSI RENDSZEREK (ERP) ÁTTEKINTÉSE, SAP

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter

SZÉCHENYI  2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

- Enterprise Resource Planning

Enterprise resource planning (ERP) is business process management software that allows an organization to use a system of integrated applications to manage the business and automate many back office functions related to technology, services and human resources.

What is Enterprise Resource Planning (ERP)? Webopedia

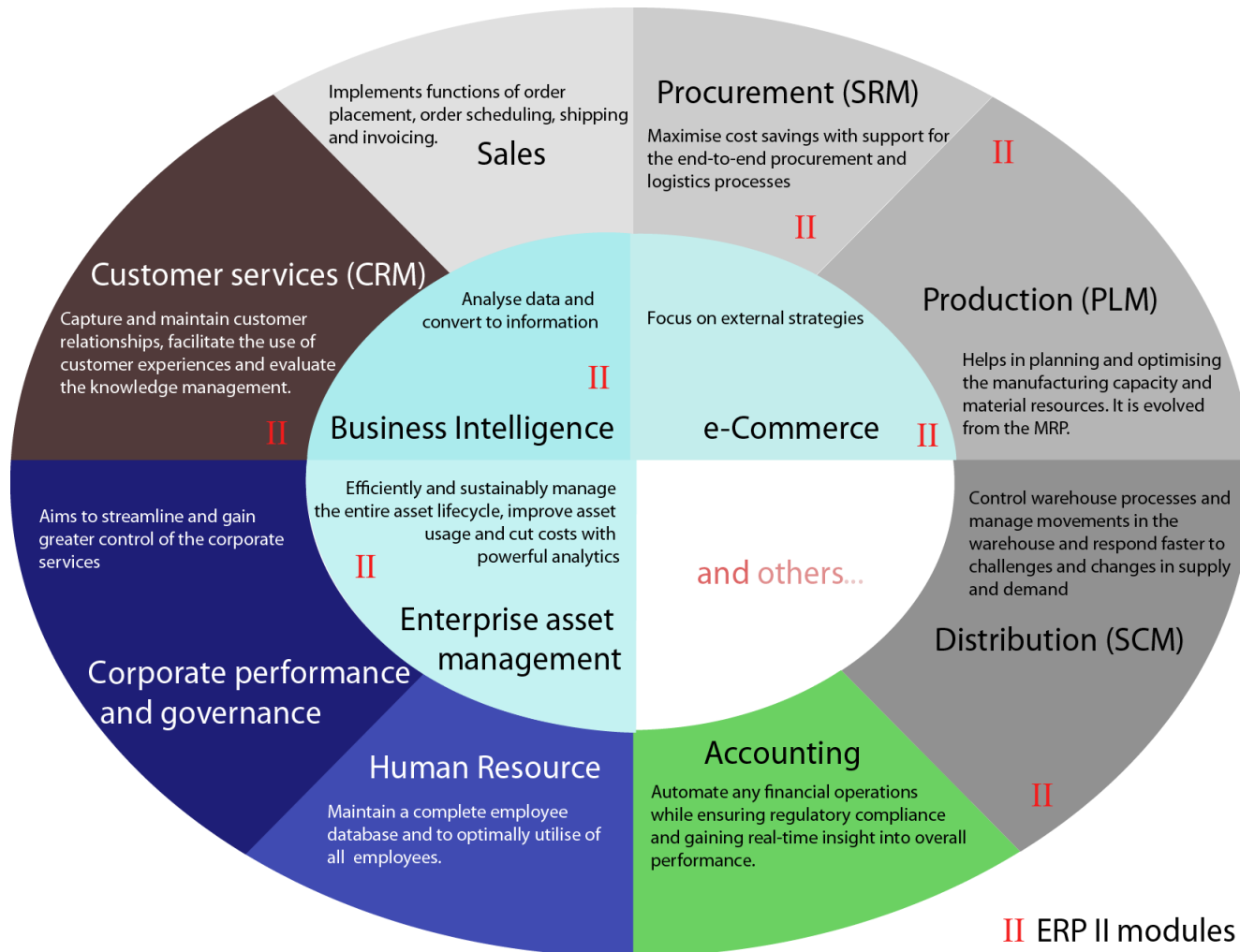
www.webopedia.com/TERM/E/ERP.html



ERP TÖRTÉNETE

- 1970-es évek: nincs még ERP rendszer
 - SAP: R/1 rendszer: valós idejű feldolgozás, központi adatbázis
- 1980-as évek: standard ERP rendszerek
 - SAP: piacvezető, integrált folyamatok - hatékonyság
- 1990-es évek: internet, E-business
 - SAP: új dimenziós termékek: CRM, APO, EPB, SEM, BW
 - vállalatokon átívelő folyamatok
- 2000-es évek: integrált Business Suite és xApps
 - E-business az ERP rendszer bővítése
 - Termékek kombinációja folyamatok támogatására

ERP TIPIKUS MODULJAI



ERP RENDSZEREK JELLEMZŐI

- Egy integrált, folytonos képet adnak az alapvető üzleti folyamatokról
- Nyomon követik
 - Üzleti erőforrásokat
 - Pénz
 - Nyersanyag
 - Termelési kapacitás
 - Stb.
 - Megrendeléseket
 - Kifizetéseket
 - Stb.
- Tipikusan
 - Integrált rendszerek közel valós idejű működéssel
 - Központi adatbázissal rendelkeznek
 - Egységes megjelenés modulok között is

ERP ELŐNYEI/HÁTRÁNYAI

- Előnyök
 - Az üzleti folyamatok integrálása gyorsítja a döntéshozatalt, csökkenti a hibalehetőségeket
 - Az adatok szervezet szinten is láthatóak
 - Nem kell több különböző rendszer adatait szinkronizálni
 - Elnevezési konvenciókat hoz
 - Segít az érzékeny adatok védelmében
- Hátrányok
 - Testreszabhatóság általában nehézkes
 - Az üzleti folyamatok ERP rendszerhez alakítása néha káros lehet
 - A teljesen független üzleti folyamatok integrálása felesleges
 - Nagy betanulási idő
 - ERP rendszerek harmonizálása hatalmas feladat

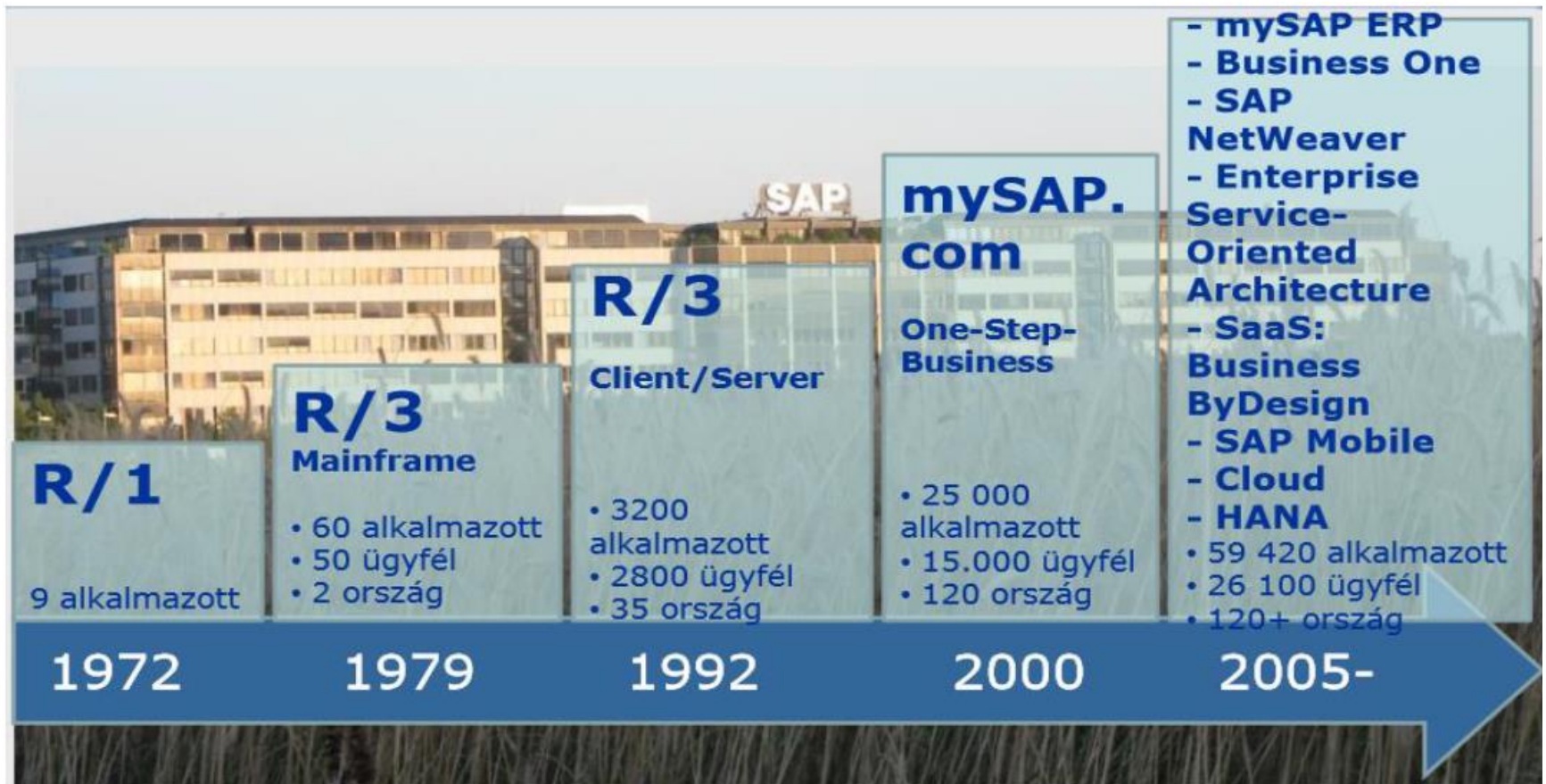
MI AZ AZ SAP?

- Egy cég?
- Egy üzleti/ERP alkalmazás?
- Egy üzleti alkalmazások megvalósítását támogató keretrendszer?
- Egy üzleti alkalmazások integrálását támogató keretrendszer?
- Egy vizuális/4GL nyelvet támogató platform?

MOTIVÁCIÓ



SAP TÖRTÉNETE (RÖVIDEN)



SAP TÖRTÉNETE (RÖVIDEN)

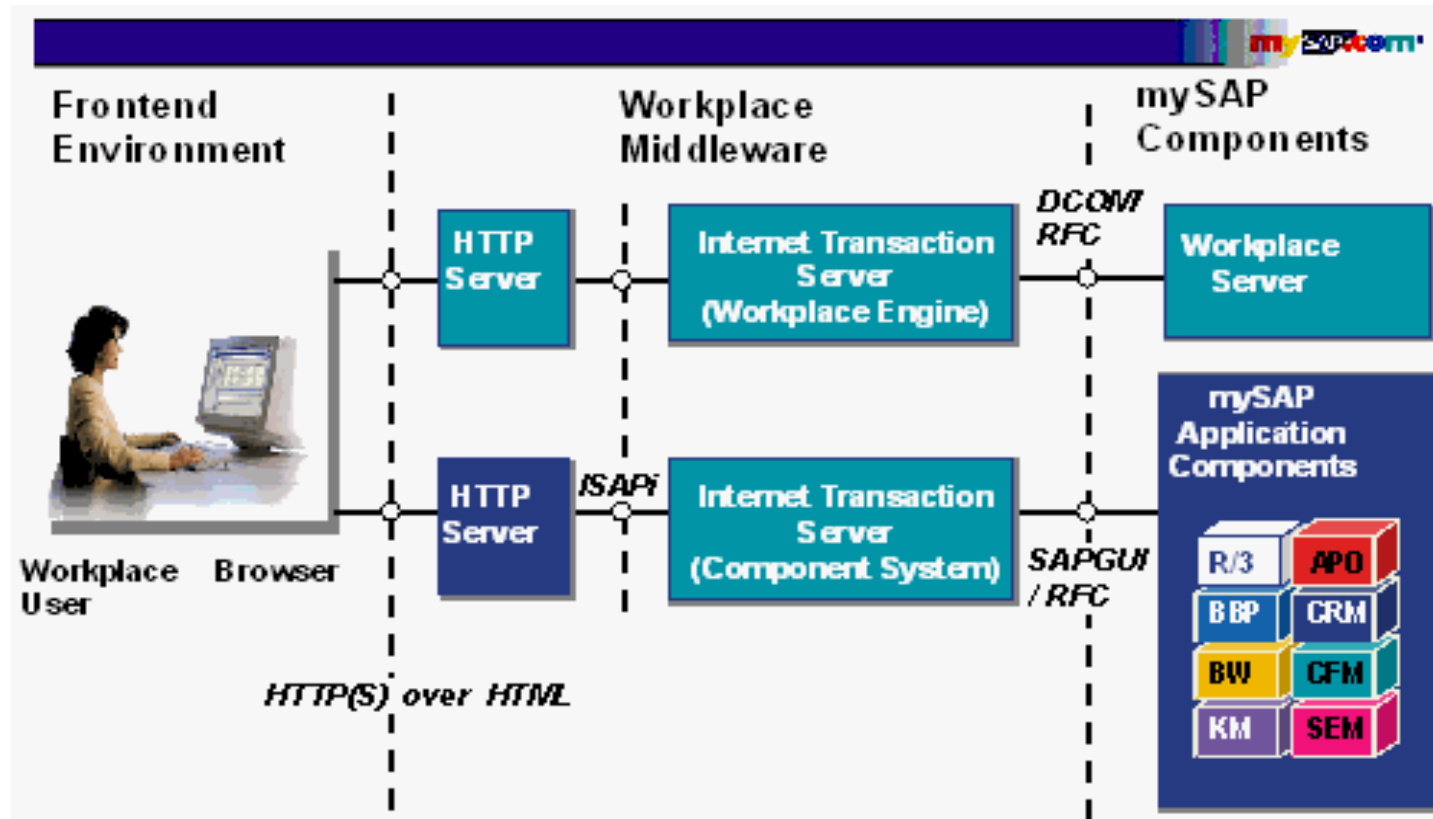
- 1972: öt korábbi IBM alkalmazott alapította
 - SAP: System Analysis and Program Development
 - Vízió: standard alkalmazás szoftver fejlesztése valós idejű üzleti feldolgozáshoz
 - 1 éven belül 9 alkalmazott, 620.000 DM forgalom
- 1973: RF - első pénzügyi könyvelés szoftver
 - R/1 rendszer alapjai
 - Első ügyfelek: Rothandle - cigaretta, Knoll – gyógyszer
 - IBM számítógépek, DOS operációs rendszer: 9 alkalmazott, 620.000 DM forgalom
- 1976: SAP GmbH megalapítása

SAP TÖRTÉNETE (RÖVIDEN)

- 1980-as években vált sikeressé az SAP R/2-nek köszönhetően
- Az SAP R/3 1992-es bevezetése után az SAP AG a világ vezető standard alkalmazási szoftver terjesztő cége lett
- 1998-ban az SAP egytermékes vállalatból globális méretű, üzleti megoldásokat szállító céggé vált
- 1999 – mySAP.com stratégia
 - nyílt, rugalmas és széles körű e-üzleti megoldásokból felépülő környezet
- 2002 - NetWeaver

MYSAP.COM ÉS NETWEAVER

- Mi az a mySAP?



MYSAP.COM ÉS NETWEAVER

- NetWeaver
 - Egy webalapú integrációs és alkalmazási platform
 - mySAP.com stratégia megvalósítása
- mySAP.com-on belül a cégek megtervezhetik saját vállalati politikájukat
- Egyedi internetes és webalapú alkalmazásokat fejleszthetnek, integrálhatnak

- SAP számos előre legyártott modullal rendelkezik
 - FI: Pénzügy és számvitel
 - CO: Kontrolling
 - EC: Vállalati controlling
 - IM: Beruházás menedzsment
 - TR: Pénzügyi menedzsment

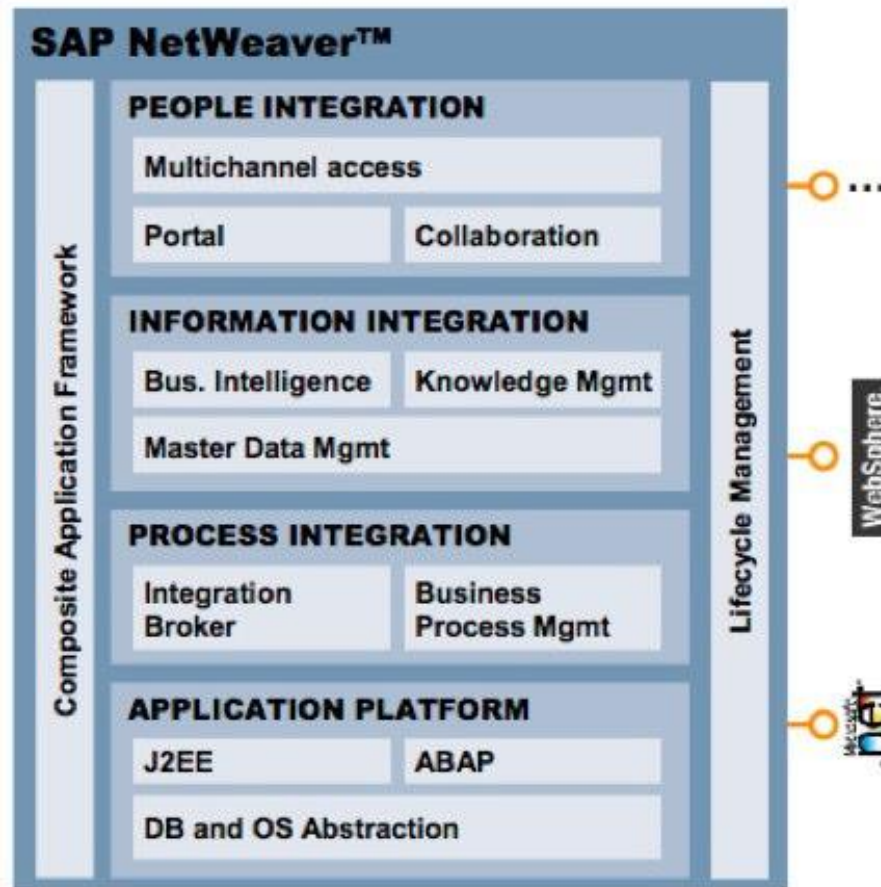
- FI: Financial accounting
 - Általános könyvelés és pénzügyi információk
 - Valós idejű mérleg, profit/veszteség listák
- Modulok:
 - FI-AA: Asset accounting (tárgyi eszköz)
 - FI-AP: Accounts payable (szállítók)
 - FI-AR: Accounts receivable (vevők)
 - FI-GL: General ledger accounting
 - FI-LC: Consolidations
 - FI-SL: Special purpose ledger

- CO: Controlling
 - Költség struktúrák és befolyásoló tényezők
 - Különböző értékelés stratégiák és mennyiségi struktúrák a termék költség számításához
- Modulok:
 - CO-ABC: Activity-based costing
 - CO-OM: Overhead cost control
 - CO-PA: Sales and profitability analysis
 - CO-PC: Product cost controlling

- EC: Enterprise controlling
 - Döntéshozzájáruló eszköz
 - EC-EIS: Executive Information System
- Legfontosabb adatok riport és grafikus formában
 - EC-MC: Management Consolidation
- Leányvállalatok adatai különböző országokból
 - EC-PCA: Profit center accounting
- Profit analízis a cég különböző üzleti területeiről

- IM: Capital Investment Management
 - Beruházási projektek és pénzügyi keretek
- TR: Treasury
 - Készpénz menedzsment és előrejelzés
- Modulok:
 - TR-CM: Cash management
 - TR-FM: Funds management
 - TR-TM: Treasury management
 - TR-MRM: Market Risk Management

SAP ALKALMAZÁS ÉS INTEGRÁCIÓS PLATFORM



- A legújabb SAP üzleti csomag
 - 2015
- SAP HANA platformra írták
 - In-memory relációs adatbázis
- Felhő támogatás
 - Kisebb adattárolási igény a helyszínen
 - Nagyobb átvitel
 - Gyorsabb elemezhetőség
 - Gyorsabb adatelérés

SAP TERMÉKPORTFÓLIÓ

- SAP Analytics
- SAP Advanced Business Application Programming (ABAP)
- SAP ERP
- SAP HANA
- SAP Portal
- SAP S/4HANA
- SAP S/4HANA Cloud
- https://en.wikipedia.org/wiki/List_of_SAP_products

- Advanced Business Application Programming
 - 4GL nyelv
 - Eredetileg az SAP R/2 riport lekérdező nyelve
 - Saját SAP riportok/komponensek fejlesztése
 - Nincs külön forrásfájl, SAP DB tárolja
 - Bytecode generálódik belőle -> futtatókörnyezet
- NetWeaver
 - Támogatja az ABAP és Java nyelveket is

ABAP HELLOWORLD

"Hello World"

```
REPORT TEST.  
WRITE 'Hello World'.
```

ABAP PROGRAMOK FAJTÁI

- Riport
 - Felhasználói input + adatbázis
 - Interaktív riport a kimenet
- Module pool
 - Összetettebb felhasználói interakció
 - Felületek (screen)
 - Flow logic
 - Process Before Output
 - Process After Input
 - Kettő együtt
 - Dynpro (dynamic program)

ABAP NEM VÉGREHAJTHATÓ PROGRAMOK

- INCLUDE modul-ok
 - Betölthető modulok
- Subroutine pool
 - Szubrutinok
- Function group
 - Önálló funkcionalitás
- Object class
- Interface
- Type pool

ABAP KOMPLEXEBB PÉLDA

```
DATA LAST_EOM    TYPE D.  "last end-of-month date

* Start from today's date
LAST_EOM = SY-DATUM.
* Set characters 6 and 7 (0-relative) of the YYYYMMDD string to "01",
* giving the first day of the current month
LAST_EOM+6(2) = '01'.
* Subtract one day
LAST_EOM = LAST_EOM - 1.

WRITE: 'Last day of previous month was', LAST_EOM.
```

KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

A CLEAN CODE PROGRAMOZÁSI ELVEK ÉS REFAKTORÁLÁS A GYAKORLATBAN

Vállalati Információs Rendszerek
© 2018-2019, Dr. Hegedűs Péter



MAGYARORSZÁG
KORMÁNYA

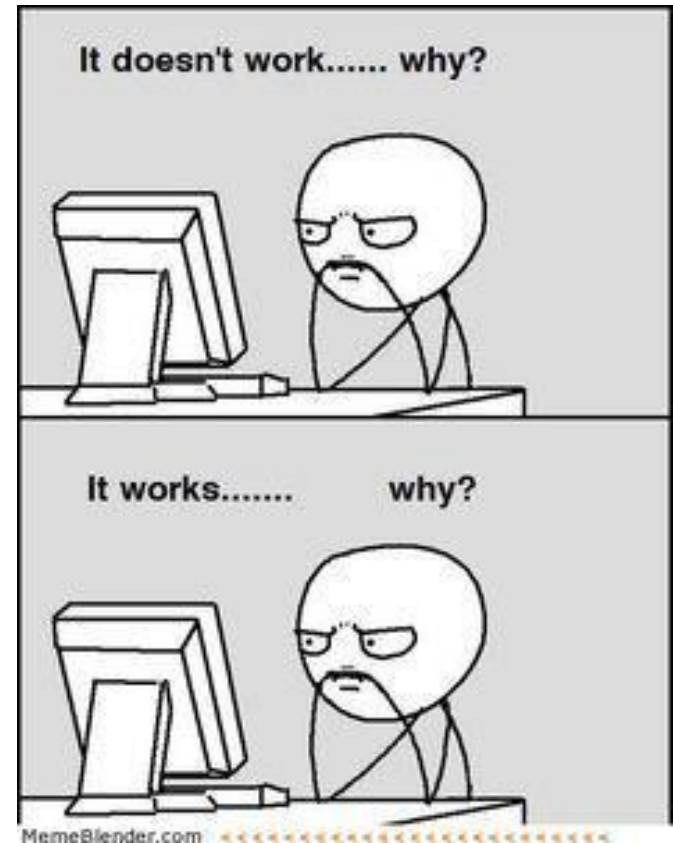
Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

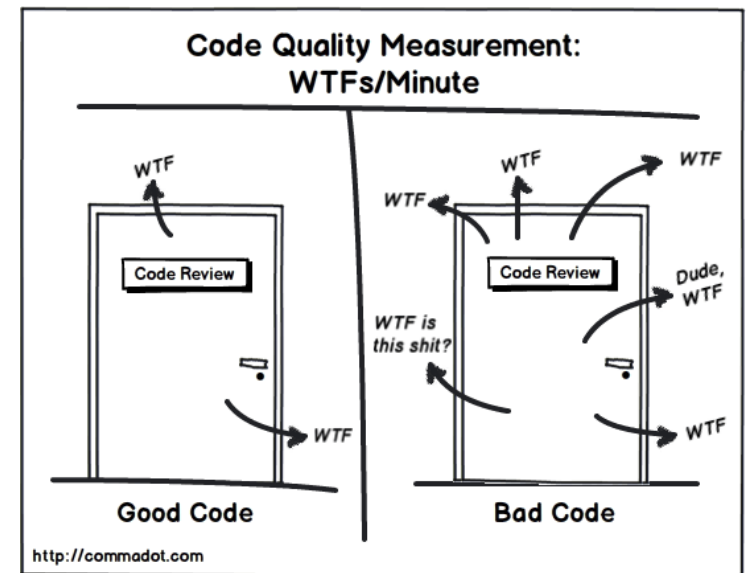
CLEAN CODE - MI AZ?

- Ahány programozó, annyi féle megfogalmazás
 - Robert C. Martin – Clean Code
- Általánosságban
 - Könnyű olvasni
 - Végrehajtás folyamatot
 - Objektumok kapcsolatát
 - Szerepét egy adott osztálynak
 - Feladatát egy adott metódusnak
 - Könnyű módosítani
 - Osztályok és metódusok “kicsik”
 - Megjósolható a viselkedésűek
 - Osztályok tömör és tiszta API-val rendelkeznek
 - Könnyű tesztelni



MIÉRT JÓ?

- Mert könnyű olvasni és megérteni 😊
- Szoftverfejlesztés -> Csapatmunka
 - A kód sohasem egyszer íródik
 - Mások és saját munkánkat tudjuk vele könnyíteni
- Könnyű megérteni
 - Fejlesztési idő csökken
- Könnyű tesztelni
 - Tesztelési idő csökken
- Könnyű bug-ot javítani és élvezhetőbb lesz a munka



HOGYAN? – ELNEVEZÉSEK (1)

- Találékony nevek
 - *Rossz: int d; //time in days*
 - *Jó: int days;*
- Könnyen olvasható nevek
 - *Rossz: long tInMls;*
 - *Jó: long timeInMilliseconds;*
- Konzisztencia
 - *Rossz: getSize(); retrieveWeight();*
 - *Jó: getSize(); getWeight();*
- Nevek hossza
 - *Rossz: List<Customer> theCustomerListWithoutFilter;*
 - *Jó: List<Customer> allCustomers;*

HOGYAN? – ELNEVEZÉSEK (2)

- Általános kódolási konvenciók

Rossz:

```
const int max = 1;  
  
public interface Customer  
public class personaddress  
void getallcustomers()
```

Jó:

```
const int MAX = 1;  
  
public interface ICustomer  
public class PersonAddress  
void getAllCustomers()
```

- Kommentek

- Rossz kódot nem teszi jobbá a comment
 - érdemes újraírni
- Dokumentációhoz, illetve fejlesztéshez/debughoz jól jön

HOGYAN? - ALAPELVEK

- DRY
- KISS
- YAGNI
- SOLID

HOGYAN? - DRY

- **Don't repeat yourself!**
 - Célja csökkenteni a kódnak az ismétlődését
 - Ha javítani kell, akkor csak egy helyen kell
 - Szervezzük ki metódusként, tegyük globálissá stb.

HOGYAN? - KISS

- **K**ee**p** **i**t **s**imple and **s**traightforward
- **K**ee**p** **i**t **s**imple, **s**tupid
- **K**ee**p** **i**t **s**hort and **s**imple
 - Az egyszerűbb jobb, kerüljük a komplexitást

HOGYAN? - YAGNI

- **You aren't gonna need it**
 - Mindig annyit implementálj, amennyi szükséges
 - Ne többet, de ne is kevesebbet
 - Több kód, több hibalehetőség -> felesleges

HOGYAN? - **S**OLID

- Egy felelősség elve (**S**ingle responsibility)
 - “*Egy osztály vagy modul egy, és csak egy felelősséggel rendelkezzen*”
 - „God” osztályok elkerülése

```
class Customer
{
    ...
    public void add()
    {
        // Database code goes here
    }
}
```

HOGYAN? - SOLID

- Nyílt/zárt elv (Open/closed principle)
 - “Egy osztály, vagy modul, legyen nyílt a kiterjesztésre, de zárt a módosításra.”

```
abstract class Employee {  
    const TYPE_TEACHER = 0;  
    const TYPE_CLEANER = 1;  
    public void setType(int type){}  
    public int getType(){}  
}
```

```
class Teacher extends Employee {  
    public Teacher() {  
        setType(TYPE_TEACHER);  
    }  
}
```

```
class Cleaner extends Employee {  
    public Cleaner() {  
        setType(TYPE_CLEANER);  
    }  
}
```

HOGYAN? - SOLID

- Liskov helyettesítési elv (Liskov substitution)
 - “*Minden osztály legyen helyettesíthető a leszármazott osztályával anélkül, hogy a program helyes működése megváltozna.*”
 - Minden négyzet téglalap...

```
class Rectangle {  
    public void setWidth(double width){}  
    public void setHeight(double height){}  
    public double getWidth(){}  
    public double getHeight(){}  
}
```

```
class Square extends Rectangle {  
    }  
  
    public void f(Rectangle r) {  
        r.setWidth(32);  
    }
```

HOGYAN? - SOLID

- Interface elválasztási elv (Interface segregation)
 - *“Több specifikus interface jobb, mint egy általános.”*

```
interface PrinterInterface {  
    public void print(Document document);  
    public Document scan();  
}
```

```
class MyReallyOldPrinter implements PrinterInterface {  
    public void print(Document document) {  
        //print document here  
    }  
    public Document scan() {  
        throw new NotImplementedException();  
    }  
}
```

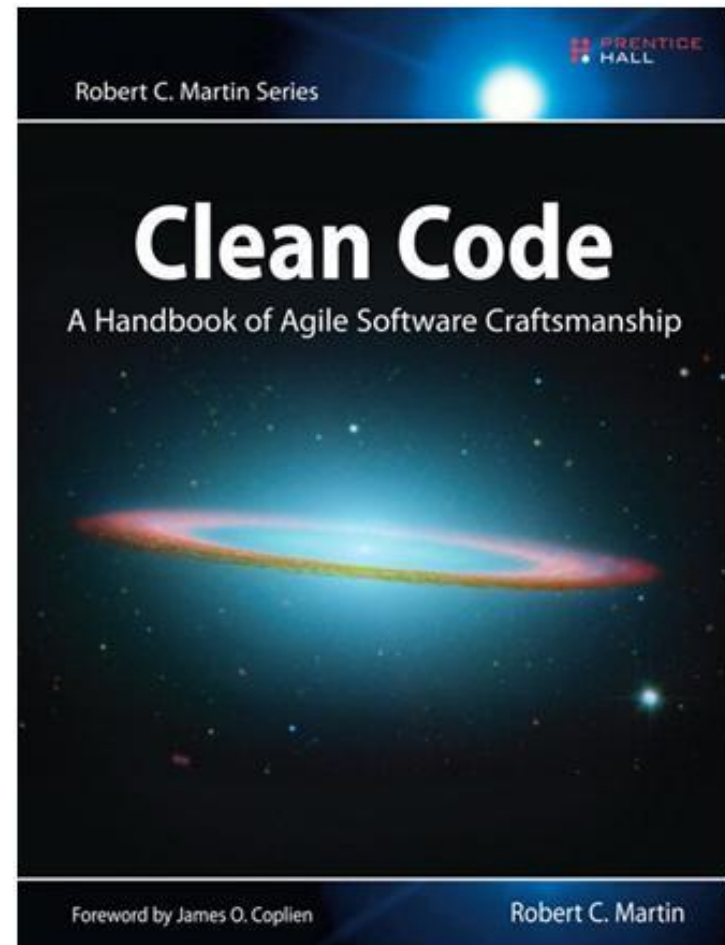
HOGYAN? - SOLID

- Függőség megfordítási elv (Dependency inversion)
 - “A kód függjön absztrakcióktól, ne konkrét implementációktól.”

```
class CustomerDAO {  
    public List<Customer> getAll() {  
        // Database code goes here  
    }  
}
```

```
class CustomerController {  
    public void usefulMethod() {  
        CustomerDAO dao = new CustomerDAO();  
        List<Customer> customers = dao.getAll();  
        ...  
    }  
}
```

Robert C. Martin – Clean Code



REFAKTORÁLÁS BEVEZETŐ

- Szoftvererózió
- Új funkciók implementálásával egyre csökken a minőség
 - kiszámíthatatlan működés
 - nehezen karbantartható kód



- Refaktoring
 - jobb minőség, karbantarthatóbb kód
 - változatlan működés

REFAKTORINGOK

```
if feltétel:  
    if 'leírás' in szótár:  
        leírás = szótár['leírás']  
    else:  
        leírás = None  
else:  
    leírás = name
```



```
leírás = name  
if feltétel:  
    if 'leírás' in szótár:  
        leírás = szótár['leírás']  
    else:  
        leírás = None  
else:  
    leírás = name  
leírás = szótár.get('leírás')
```



MOTIVÁCIÓ

- Refaktoring pozitív hatása elfogadott
 - Azonban kevés objektív, empirikus kutatási eredmény
 - Sokszor zárt forrású, ipari programokat elemeznek
 - Nem reprodukálható
 - Nem általánosítható
 - Nyílt forrású kódok elemzésénél pedig a kis és közepes rendszerek dominálnak
- Célunk egy nagy méretű, iparban is használt rendszeren a refaktoring minőségre gyakorolt hatásának vizsgálata

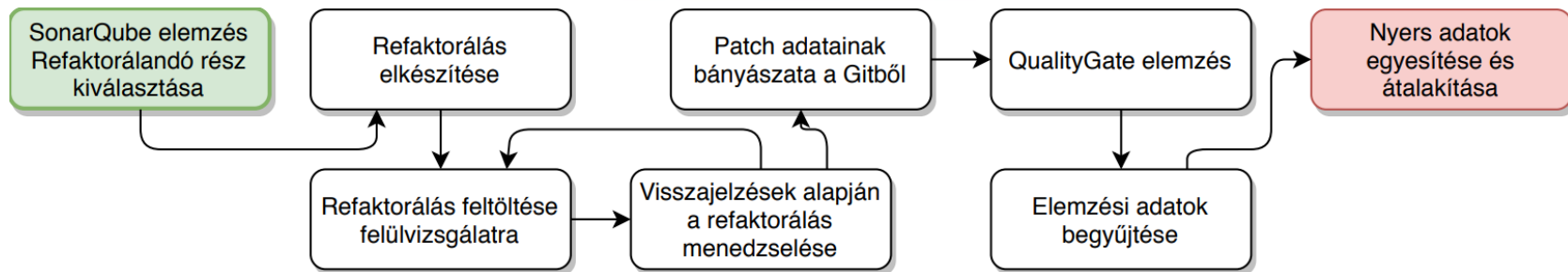
KÓDMINŐSÉG ÉS MÉRÉSE

- Refaktoring definíciója „egzakt”, de hogyan mérjük a minőséget?
- ISO/IEC 25010
 - Funkcionalitás, megbízhatóság, használhatóság, hatékonyság, hordozhatóság, karbantarthatóság
- QualityGate
 - Statikus forráskód-elemző és minősítő rendszer
- SonarQube (SourceMeter plugin)
 - Bad smellek, sebezhetőségek, hibák, duplikációk detektálása és tesztlefedettség mérése



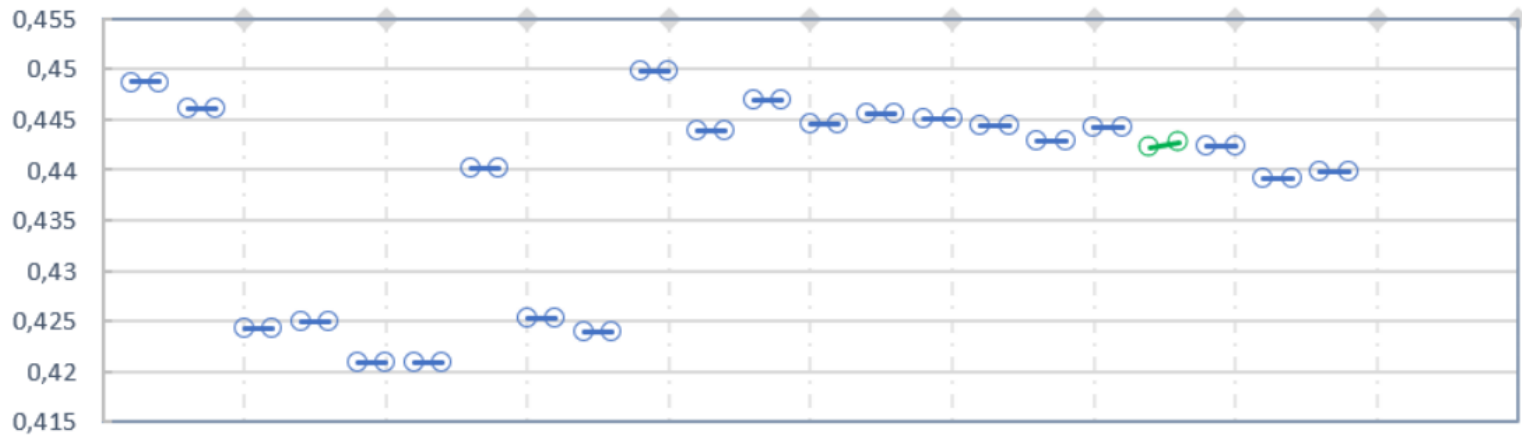
- Együttműködés
 - Ericsson Magyarország Kft.
 - SZTE Szoftverfejlesztés Tanszék
- Nyílt forráskódú felhő rendszer
 - Összesen 2+ millió Python kódsor
 - 650+ vállalkozás és közel 90.000 fejlesztő
- Nova
 - 363.000 sor, 87% tesztlefedettség
- Watcher
 - 44.000 sor, 86% tesztlefedettség.

MÓDSZERTAN

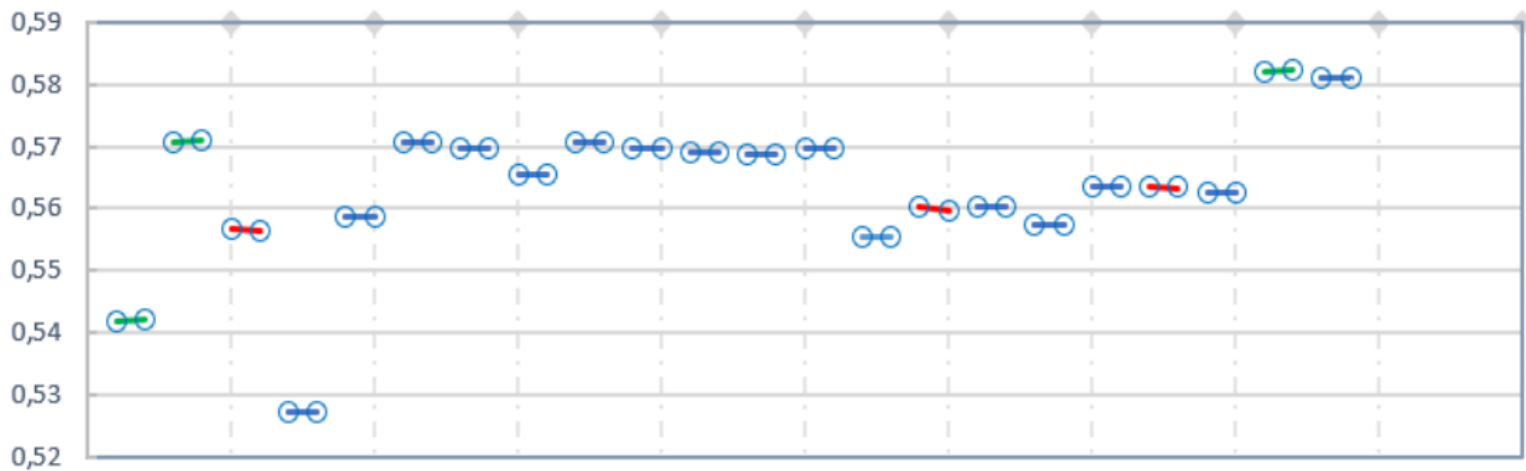


- Refaktoring elkészítése, menedzselése
- 47 db elfogadott (Nova 22, Watcher 25)
- QualityGate elemzés
- Nyers adatok kinyerése az adatbázisból
- Adatok átalakítása olvashatóbb formára
- Adatok feldolgozása

EREDMÉNYEK VIZSGÁLATA

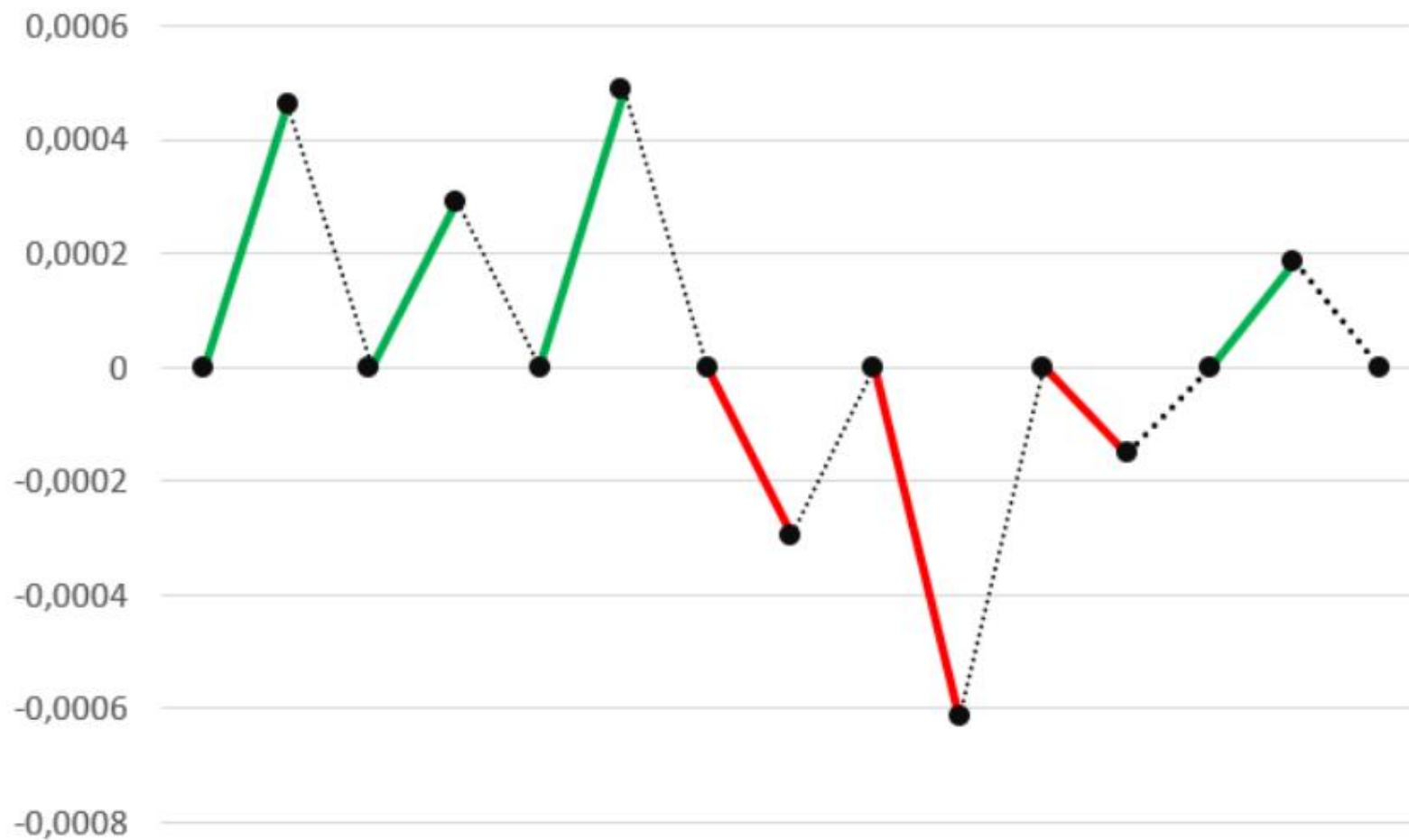


Nova modul (21/1/0)



Watcher modul (19/3/3)

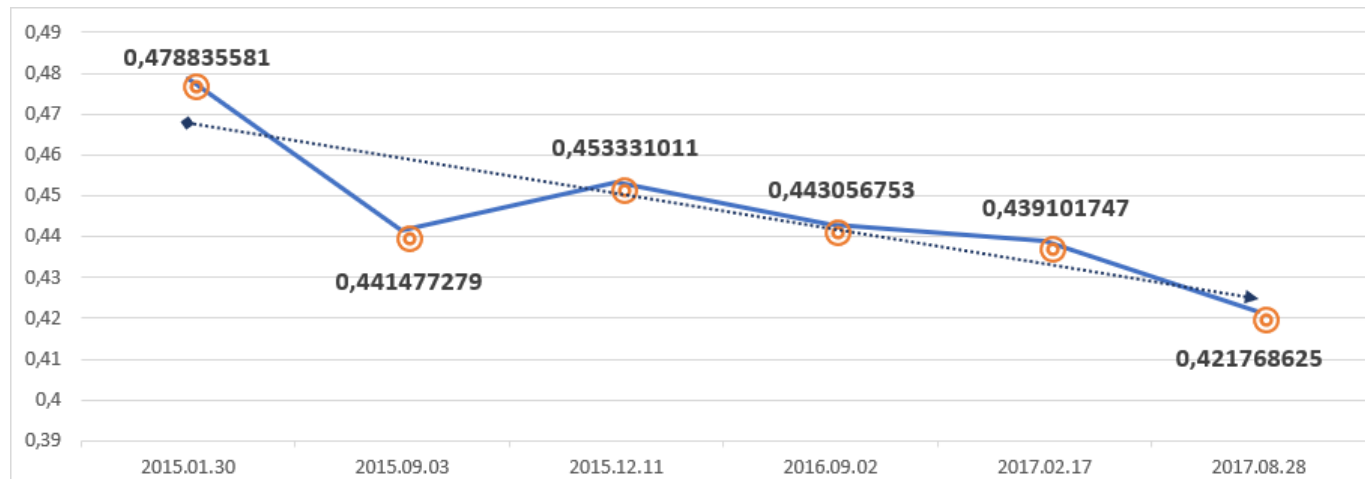
EREDMÉNYEK VIZSGÁLATA



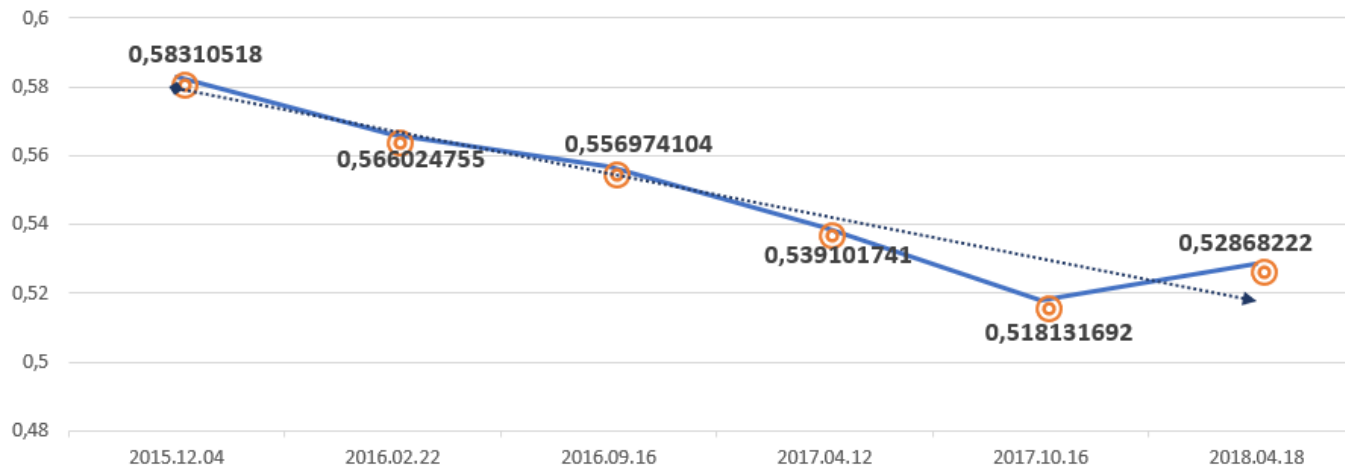
EREDMÉNYEK VIZSGÁLATA

- 1 nagy és komplex metódus – 5 kisebb
 - Komplexitás, hibatűrés, stabilitás, tesztelhetőség
 - Elemezhetőség, változtathatóság, paraméterek
- Ősosztálybeli konstruktorhívások pótlása
 - Nem volt kapcsolat ős- és gyermekosztály között
 - Rossz irányba mozgatja a minőségmutatókat
- Kód duplikáció feloldása – 1 fájl törölve
 - A fájl egyszerű, a függvény pedig kicsi és jó minőségű volt
 - A kód minősége rendszer szinten így csökkent

TREND VIZSGÁLATA



Nova modul trendvonal



Watcher modul trendvonal

SAJÁT EREDMÉNYEIM

- Közreműködés ~35 refaktoring elkészülésében
- Az adatbányászathoz erre alkalmas céleszközök elkészítése
- Elemzés felügyelete
- Eredmények vizsgálata, trendanalízis
- Minőségváltozások kézi ellenőrzése

KONKLÚZIÓ

- Kicsi, de pozitív változás, hasznosak
 - Egyik mutató nő, a másik romlik
- Nagy rendszer – nehéz kimutatni
- Folyamatos refaktorálás hosszú távon hozhat látványos javulást
 - „Sok kicsi sokra megy”
- Jövőbeli tervek
 - A többi modulra is kiterjeszteni a kutatást

KÖSZÖNÖM A FIGYELMET!

*„A tananyag az EFOP-3.5.1-16-2017-00004
pályázat támogatásával készült.”*

SZÉCHENYI 2020



MAGYARORSZÁG
KORMÁNYA

Európai Unió
Európai Szociális
Alap



BEFEKTETÉS A JÖVŐBE

KAPCSOLÓDÓ E-FELADAT GYŰJTEMÉNY

- A Vállalati Információs Rendszerek tárgy gyakorlatához egy e-feladat gyűjtemény is kapcsolódik, amelyet az alábbi linken lehet elérni

http://www.inf.u-szeged.hu/~hpeter/pages/VIR_efeladat