

**Eötvös Loránd Tudományegyetem
Informatikai Kar**

Szoftvertchnológia

9. előadás

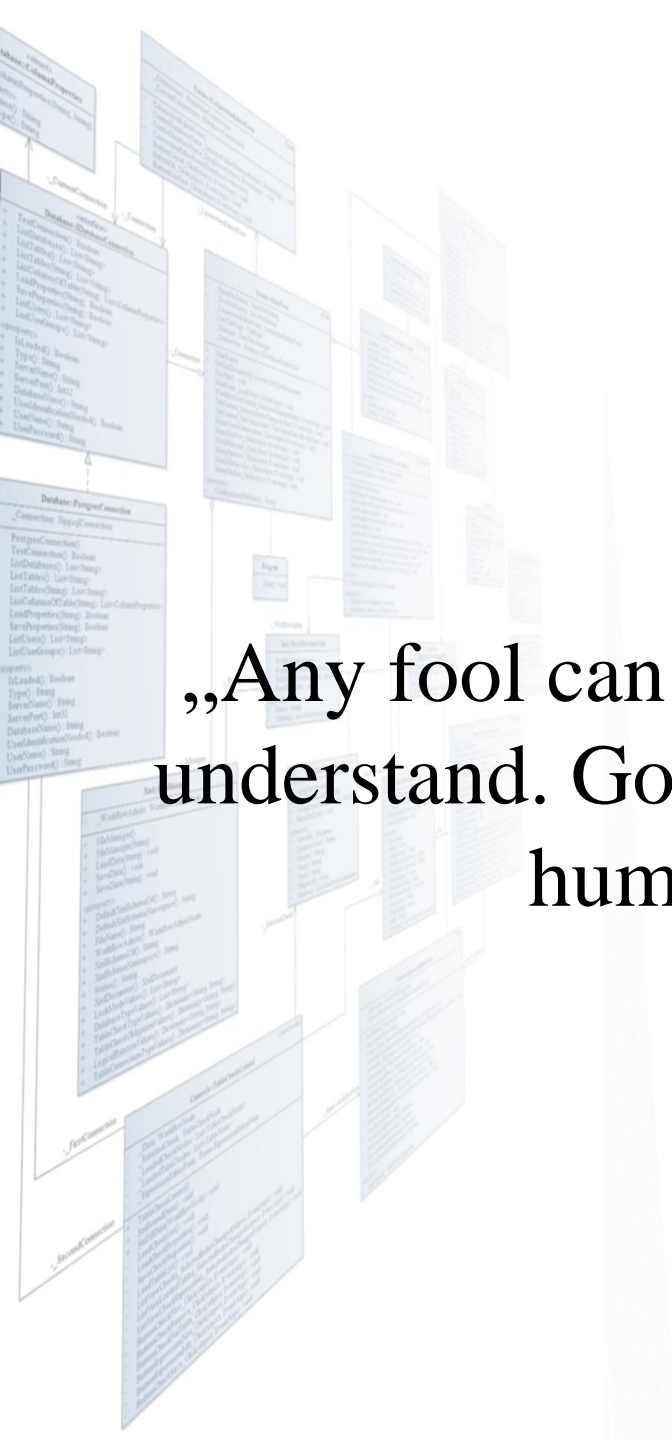
Implementáció és verziókövetés

Giachetta Roberto

groberto@inf.elte.hu

<http://people.inf.elte.hu/groberto>





„Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

(Martin Fowler)

Implementáció és verziókövetés

Implementáció

- Az implementáció a programkód elkészítése a rendszertervnek megfelelően adott programozási nyelven
 - az adatok megfelelő ábrázolása, reprezentálása
 - a funkciókat megvalósító algoritmusok alkalmazása
 - korábban már bevált elemek (algoritmusok, programszerkezetek) újhasznosítása
 - a minőségi mutatóknak megfelelő optimalizálások (teljesítményjavítások) végrehajtása
- Az implementációt *verifikáció* zárja, amelyben ellenőrizzük, hogy a szoftver teljesíti-e a tervben megszabott funkciókat

Implementáció és verziókövetés

Újrahasznosítás

- Az implementáció általában nagyban támaszkodik *újrahasznosításra*
 - garantálja, hogy jó, hibamentes megoldások kerüljenek alkalmazásra
 - csökkenti az implementáció (és a tesztelés) idejét és költségeit
 - az újrahasznosítás elvégezhető objektum, csomag, vagy komponens szinten
- A fejlesztő nem csupán az általa korábban fejlesztett elemeket használhatja újra, de más fejlesztő által megvalósított elemeket
 - pl. nyílt forráskódú programcsomagok
 - általában előre fordított formában állnak rendelkezésre

Implementáció és verziókövetés

Az integrált fejlesztőkörnyezet

- Az implementációhoz megfelelő *integrált fejlesztőkörnyezet (IDE)* szükséges (pl. *Eclipse, Visual Studio, Xcode, NetBeans*)
 - a teljes szoftver életciklust támogatja, integrálja a verziókövetést és a tesztelést
 - a fejlesztést kód-kiemeléssel (*syntax highlight*), kód-kiegészítésekkel (*code-snippet, intelligent code completion*), illetve kód-újratervezési eszközökkel támogatja
 - megkönnyíti külső programcsomagok integrációját (*package manager*)
 - a tesztelést nyomkövetéssel (*debugging*), egységtesztek (*unit test*), illetve teljesítményteszteléssel támogatja

Implementáció és verziókövetés

Csapatban történő implementáció

- A szoftverek általában csapatban készülnek
 - az implementáció egy egységes kódolási stílus mentén történik, egységes eszköztárral
 - minden fejlesztő csak a saját programkódján dolgozik
 - a verziókövető rendszerben általában külön fejlesztési ágban tevékenykedik
 - amennyiben más kódjában hibát talál, hibajelzést tesz
 - az általa biztosított interfészeket csak egyeztetés után módosítja
 - dokumentálja (kommentezi), illetve teszteli a saját kódját (elkészíti a megfelelő egységteszteket)

Implementáció és verziókövetés

Kódolási stílus

- A *kódolási stílus* (*coding style*) egy szabályhalmaz, amely a forráskód megjelenésére (pl. elnevezés, indentálás, szóközök elhelyezése, ...) ad iránymutatást
 - a kódolási stílus követése javítja a kód értelmezhetőségét, a későbbi karbantartást, és lehetővé teszi a fejlesztők közötti kommunikáció zökkenőmentességét
 - a jó programozási stílus általában szubjektív, nem túl szigorú, de alapvető elemeket definiál
 - a kódolási stílus lehet nyelvi szinten, vállalati szinten, vagy szoftverszinten rögzített
 - a fejlesztőkörnyezetek általában lehetőséget adnak a forrás automatikus formázására

Implementáció és verziókövetés

Kódolási stílus

- Pl.:

```
class Point { // camel case (upper, Pascal case)
private:
    int xCoordinate; // camel case (lower)
    int yCoordinate;
    ...
    double DistanceTo(Point other) const
        // camel case (upper)
    {
        return ...
    } // K&R
    ...
};
```


Implementáció és verziókövetés

Kódolási stílus

- Pl.:

```
class Point {
private:
    int x_coordinate; // snake case
    int y_coordinate;
    ...
    double Distance_To(Point other) const {
        // Oxford case
        return ...
    } // 1TBS
    ...
};
```

Implementáció és verziókövetés

Kódolási stílus

- Pl.:

```
class Point {
private:
    int m_nXCoord; // hungarian notation
    int m_nYCoord;
    ...
    double distanceTo(Point pOther) const
        // camel case (lower)
    {
        return ...
    } // GNU
    ...
};
```

Implementáció és verziókövetés

Kódolási stílus

- Általános érvényű javaslatok:
 - kódrészletek megfelelő elválasztása (szóköz, sortörés, behúzás, függőleges igazítás)
 - beszédes és konzisztens elnevezések használata (kevesebb kommentezést igényelnek)
 - beégetett tartalmi elemek (számok, szövegek) megnehezítik a karbantartást (*hard coding*), ezért célszerű a kerülése, kiemelése fejlécbe, vagy konfigurációs fájlba (*soft coding*)
- A kódolási konvenció rákényszeríthető a programozóra kódolási stílus ellenőrző eszköz segítségével
 - pl. *C++Test*, *StyleCop*

Implementáció és verziókövetés

Kommentezés

- A kódot a stílusnak megfelelő kommenttel kell ellátni
 - alapvető fontosságú az interfész kommentezése (osztályok, függvények, paraméterek)
 - a megvalósítás kommentezése összetett funkcionalitás esetén hasznos, de megfelelő kódolási stílus esetén nem szükséges
 - tartalmazhat speciális jelöléseket (pl. **TODO**, **FIXME**)
 - a túl kevés, vagy túl sok komment is ártalmas lehet
- A kommentek felhasználhatóak dokumentáció előállítására is (pl. *Doxygen*, *Javadoc*), amennyiben azokat megfelelő séma szerint hozzuk létre

Implementáció és verziókövetés

Kommentezés

- Pl.:

```
// a type representing a 2D point
```

```
class Point {
```

```
...
```

```
// computes the Euclidean distance to another
```

```
// point
```

```
double DistanceTo(Point other) const
```

```
{
```

```
    return sqrt(pow(...) + pow(...));
```

```
}
```

```
...
```

```
};
```

Implementáció és verziókövetés

Kommentezés

- Pl.:

```
// Name: Point
//
// Purpose: This type represents a point in a 2D
// coordinate system.
// Remarks: Is based on double coordinates.
//
// License: LGPL v2.
//
// Author: Roberto Giachetta
// Date: 27/11/2014
// Contact: groberto@inf.elte.hu
...
class Point {
```

Implementáció és verziókövetés

Kommentezés

- Pl.:

```
// Name: distance
// Purpose: This method computes the Euclidean
// distance to another Point instance.
// Remarks: This a query method.
// Parameters: other : another point
// Return value: The distance to the other.
double distance(Point other) const
{
    // uses sqrt and pow functions from math.h
    // formula: ...
    return sqrt(pow(...) + pow(...));
}
...
```

Implementáció és verziókövetés

Kódolási stílus

- Általában nyílt és zárt programkódokra más szabályok vonatkoznak
 - *nyílt forráskód* esetén törekedni kell, hogy a kód minél gyorsabban értelmezhető legyen bárki számára
 - követni kell a programozási nyelv tördelési és elnevezési konvencióit
 - a kód megfelelő mennyiségű megjegyzéssel kell ellátni
 - *zárt forráskód* esetén a cél a fejlesztőcsapat minél nagyobb rálátása a kódra
 - törekedni kell, hogy minél nagyobb kódmennyiség legyen egyszerre áttekinthető (kevesebb helyköz és komment)

Implementáció és verziókövetés

Statikus kódelemezés

- A *statikus kódelemzés* (*static code analysis*) lehetővé teszi, hogy a forráskódot még a fordítás előtt előfeldolgozzuk, és a lehetséges hibákat és problémákat előre feltérképezzük
 - a fejlesztőkörnyezet beépített *kódelemző*vel rendelkezhet, amely megadott szabályhalmaz alapján a lehetséges hibaeseteket felfedi
 - a statikus kódelemzés egy része kimondottan a kódolási konvenciók (pl. elnevezések, tagolás, dokumentáltság) ellenőrzését biztosítja
 - a kódra számíthatók *metrikák* (*code metric*), amelyek megadják karbantarthatóságának, összetettségének mértékét (pl. *cyclomatic complexity*, *class coupling*)

Implementáció és verziókövetés

Kód-újratervezés

- A *kód-újratervező (refactoring)* eszközök célja, hogy a kód szerkezetét mindig konzisztens módon, a viselkedés befolyásolása nélkül tudjuk módosítani
 - a teljes újratervezést kisebb lépések (*micro-refactorings*) sorozatával érjük el
 - pl. átnevezések, ismétlődő kódok kiemelése, típuscsere, interfész, vagy ősosztály kiemelése, tervezési minta bevezetése
 - a kód nem funkcionális követelményeinek javítására szolgál, általában a karbantarthatóság, illetve a bővíthetőség növelése a cél
 - alkalmas bizonyos rejtett hibák, vagy sebezhetőségek felfedésére

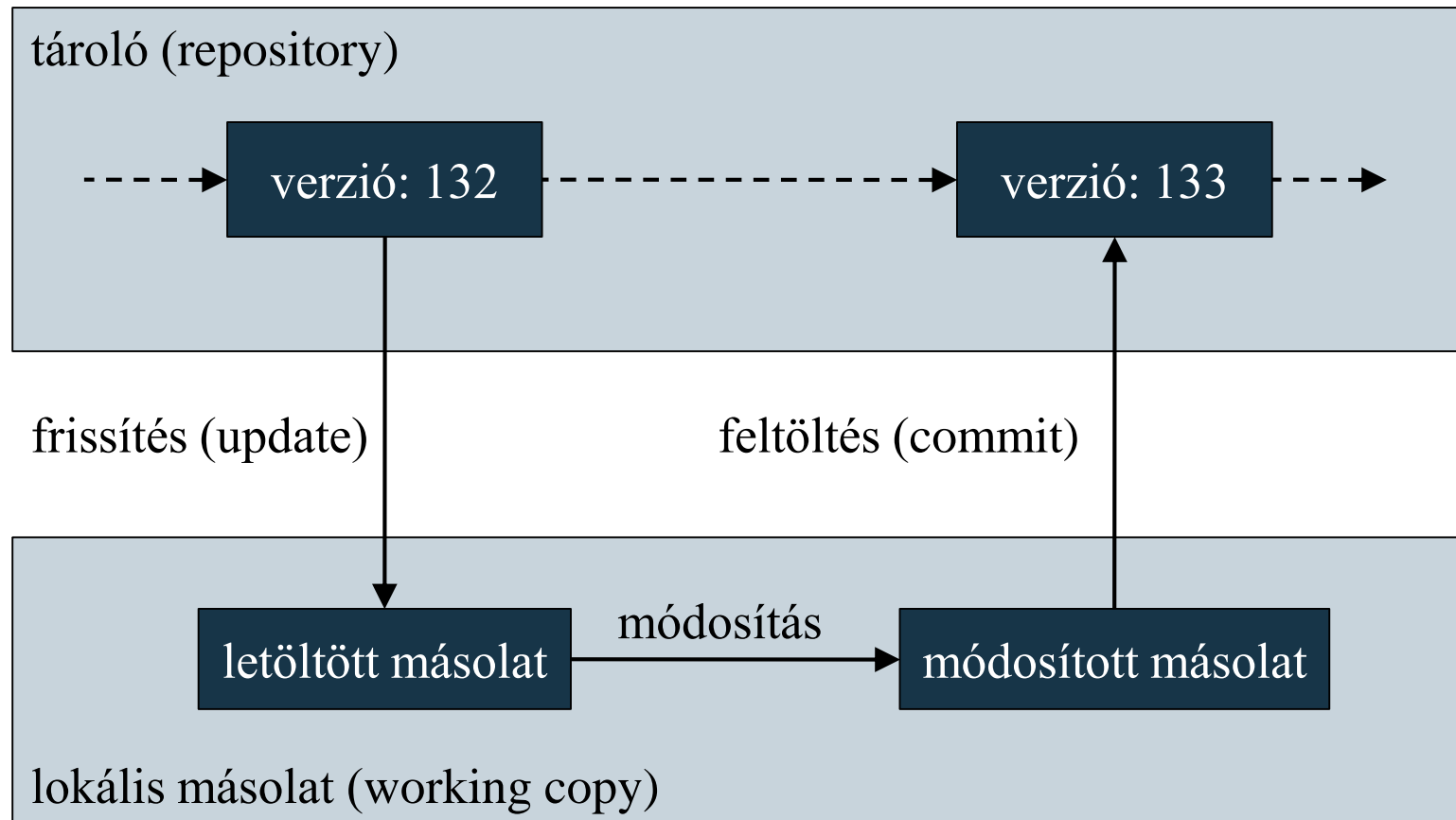
Implementáció és verziókövetés

Verziókövető rendszerek

- A verziókövető rendszerek (*revision control system*) célja a forráskód változásának követése, és a korábbi állapotok megőrzése
 - ezáltal könnyen áttekinthető, ki milyen módosításokat hajtott végre a forráson, amelyek szükségszerűen visszavonhatóak
 - egy közös tárolóban (*repository*) tartják kódokat, amit a fejlesztők lemásolnak egy helyi munkakönyvtárba (*working copy*), és ebben dolgoznak
 - a módosításokat visszatöltik a központi tárolóba (*commit*)
 - a munkakönyvtárakat az első létrehozás (*checkout*) után folyamatosan frissíteni kell (*update*)
 - pl. *Subversion* (SVN), *Git*, *Team Foundation Server* (TFS)

Implementáció és verziókövetés

Verziókövető rendszerek



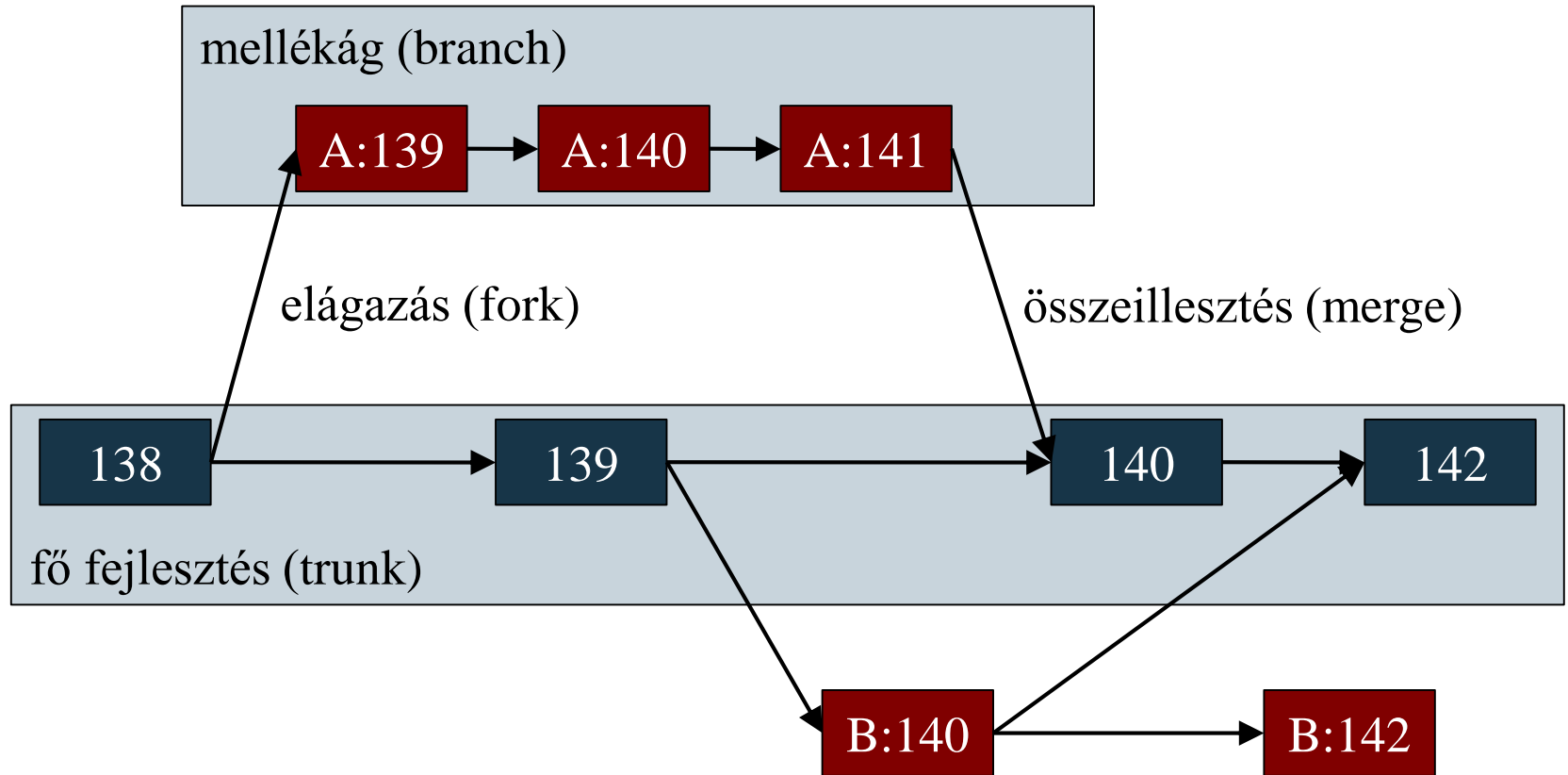
Implementáció és verziókövetés

Verziókövető rendszerek

- A rendszerek biztosítják:
 - az összes eddig változat (*revision*) eltárolását és lekérdezését, benne a legfrissebb változattal (*head*)
 - az egyes változatok közötti különbségek nyilvántartását (akár karakterek szintjén)
 - változtatások visszavonását, korábbi változatra visszatérést
 - a fő fejlesztési vonal (*baseline, master, vagy trunk*) mellett további fejlesztési vonal elágazását (*fork*), párhuzamos követését, valamint az ágak összeillesztését (*merge*)
 - a módosítások közötti konfliktusok kezelését (*resolve*)

Implementáció és verziókövetés

Verziókövető rendszerek



Implementáció és verziókövetés

Elosztott verziókövető rendszerek

- A központosított verziókezelők mellett elterjedtek az *elosztott (distributed)* rendszerek
 - nagyobb szabadságot adnak a tárolók kezelésében, a szinkronizálási lehetőségekben
 - egy főtároló (*origin*) csak egy kiindulási pont, tetszőleges másolatot készíthetünk (*clone*) belőle
 - a módosítások csak a lokális másolatot befolyásolják, külön kell feltöltenünk azokat valamely távoli tárolóra (*push*), vagy jelölhetjük őket feltöltésre (*pull request*)
 - az egyes tárolók állapotait szinkronizálhatjuk (*pull, fetch*)
 - pl. *Git, Bazaar, Mercurial*

Implementáció és verziókövetés

Elosztott verziókövető rendszerek

