



Kód átvizsgálás

(Code review)



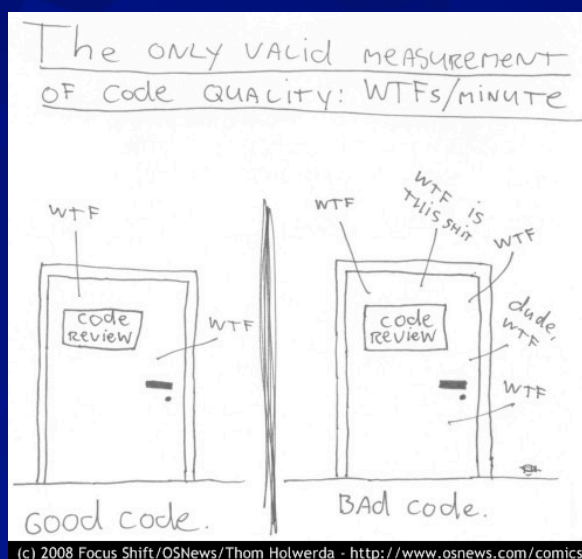
Irodalom

- ☀ Jason Cohen: Best kept secrets of peer code review, Smart Bear Inc., 2006

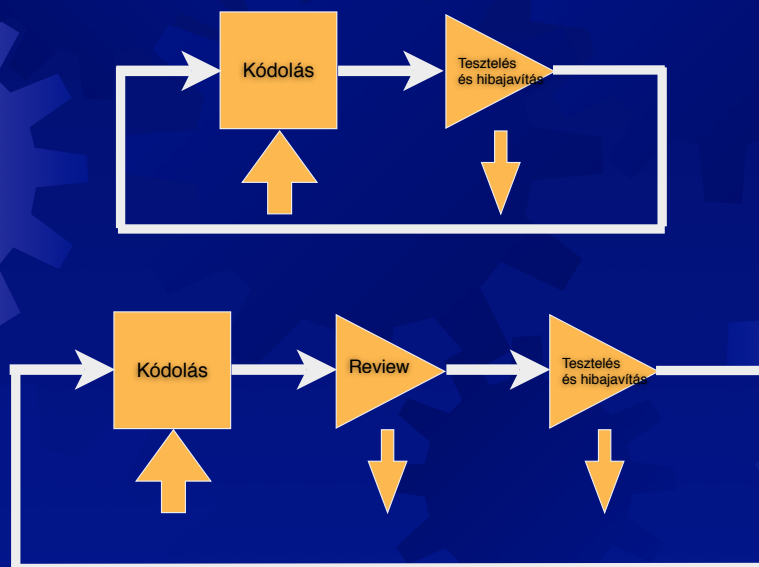
Célok, előnyök

- ☀ Jobb minőségű kód
 - jobban karbantartható
- ☀ Kevesebb hiba a kódban
 - rövidebb tesztelés
- ☀ Kóddal kapcsolatos kommunikáció javítása
- ☀ Kezdő programozók képzése

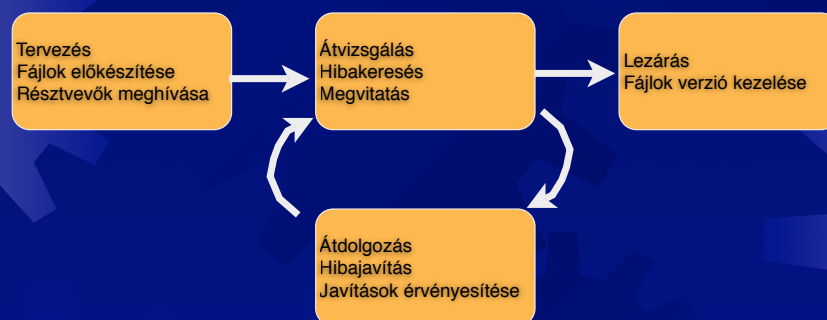
Kód minőség :)



Szoftver folyamatok



Az átvizsgálás



Az átvizsgálás (folyt.)

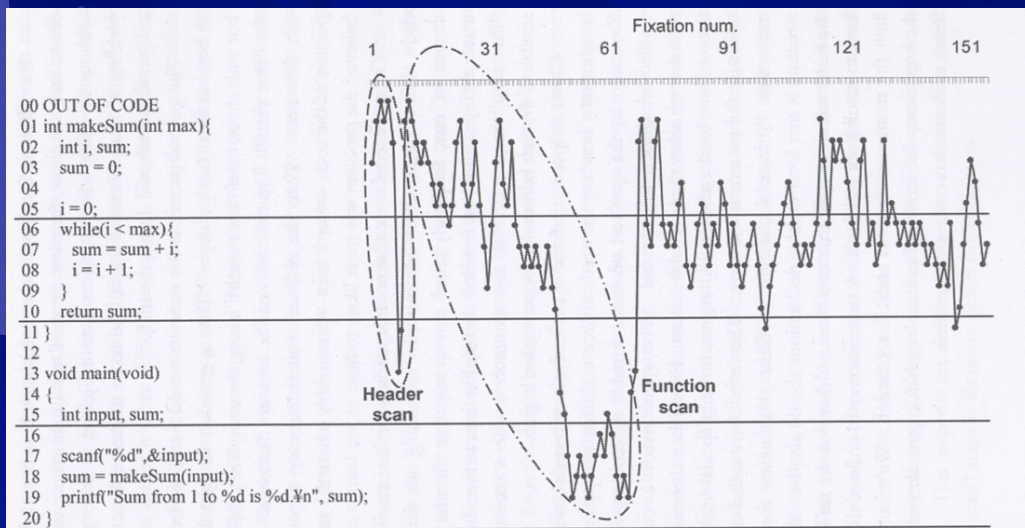
☀ Szereplők

- menedzser
 - átvizsgálás megtervezése
- moderátor
 - átvizsgálás vezetése
- szerző
 - átvizsgálendő anyagok biztosítása
- átvizsgáló
- jegyző

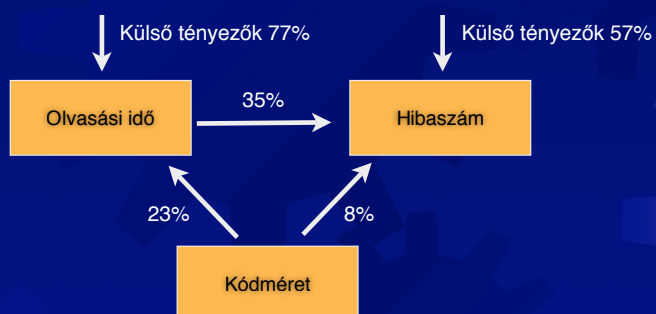
Kódolvasás

- ☀ Kezdeti “áttekintés”
 - kódsorok 80%-a
 - header scanning
- ☀ 4-5 soros koncentrált vizsgálat
- ☀ Back-tracks
 - deklarációk
 - ciklusok

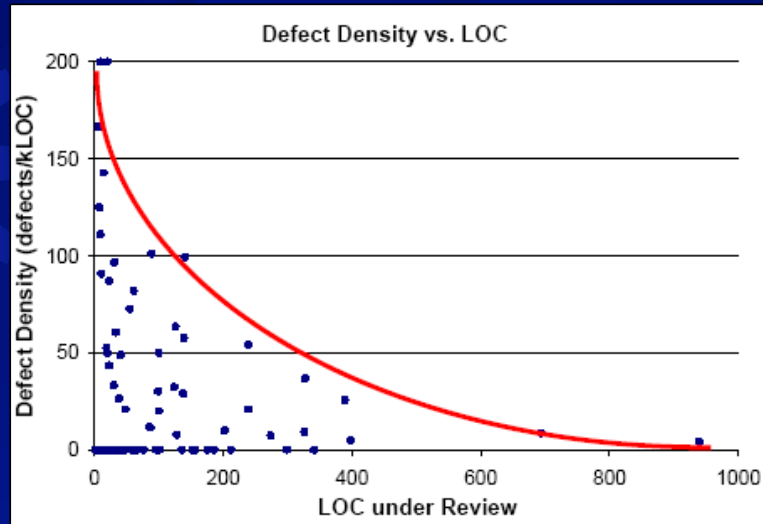
Kódozolás (folyt.)



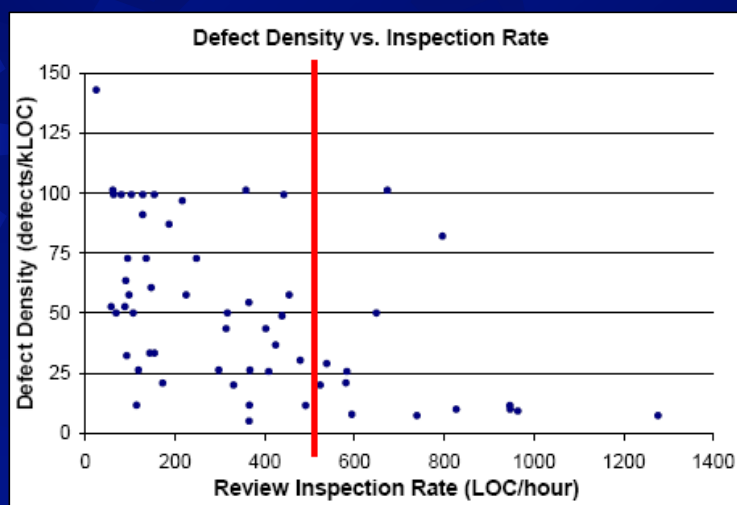
Az átvizsgálás kauzális modellje



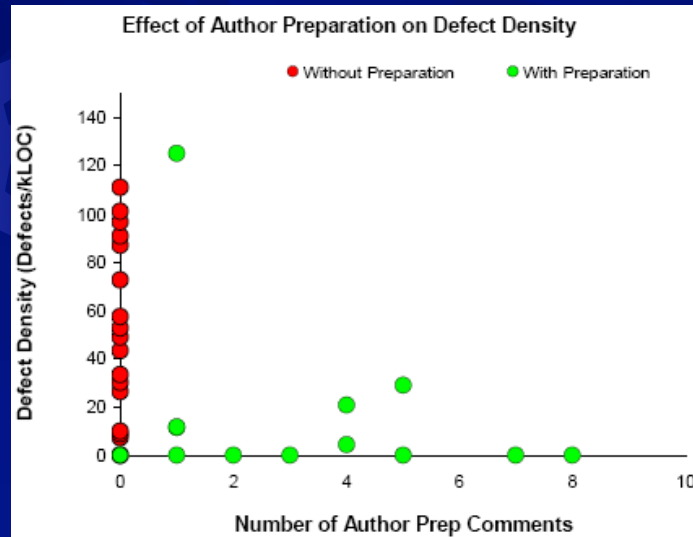
Kódméret



Átvizsgálási idő



Felkészülés az átvizsgálásra



Szociális effektusok

- ☀️ “Ego” effektus
 - hatás a fejlesztési stílusra
- ☀️ Kölcsönös tanulás
 - reviewer is!
- ☀️ Szisztematikus személyes fejlődés
- ☀️ Sértődések, Big Brother effektus
- ☀️ Tudatosítani: minél több hiba kiderül, annál jobb

Review technikák

- ☀ Formális átvizsgálás
- ☀ “Váll feletti” átvizsgálás
- ☀ E-mailben átküldéses átvizsgálás
- ☀ Eszközzel támogatott átvizsgálás
- ☀ Páros programozás

Formális átvizsgálás

- ☀ Tervezés
- ☀ Nyitó meeting
- ☀ Átvizsgálási meetingek
- ☀ Átdolgozás
- ☀ Verifikálási meeting
- ☀ Lezárás
- ☀ Követő meetingek
 - átvizsgálási eljárás javítása

“Váll feletti” átvizsgálás

- ☀ Előkészítés
- ☀ Átvizsgálás
 - fejlesztőé a vezető szerep
 - hibák feljegyzése
- ☀ Átdolgozás
- ☀ Lezárás

E-mailben átküldéses átvizsgálás

- ☀ Kód elérhetővé tétele
 - automatizált folyamat (SCM)
 - reviewerek értesítése
- ☀ Átvizsgálás
 - egyéni értékelés
 - kérdéses helyzetek feloldása
- ☀ Átdolgozás
 - követhető változtatások a fejlesztők részéről
- ☀ Lezárás

Eszközzel támogatott átvizsgálás

- ☀ Automatizált fájl kigyűjtés
- ☀ Kombinált megjelenítés
 - Változások
 - Megjegyzések, threaded kommunikáció
 - Hiba követés
- ☀ Automatizált metrika gyűjtés
 - kLOC/h
 - hiba találat/h
 - hiba/kLOC
- ☀ Integrálás IDE-kkel

Objektum orientált technikák

- ☀ Ellenőrző listás átvizsgálás
 - hibamodell alapú ellenőrzés
- ☀ Szisztematikus átvizsgálás
 - osztály hierarchia alkalmas részének átvizsgálása (előkészítés)
- ☀ Use-case átvizsgálás
 - átvizsgálás kód felhasználás alapján

Könnyű súlyú peer review

- ☀ Review előnyök megtartása, ráfordítás csökkentése
- ☀ Workflow integrálás
- ☀ Speciális eszköz támogatás
 - átvizsgálási szabályok betartása
 - metrika gyűjtés
 - jelentés generálás
 - tool integrálás

Könnyű súlyú peer review (folyt.)

- ☀ Forrás kód előkészítés egyszerűsítése
- ☀ Átvizsgálási folyamat nyomonkövetése
- ☀ Javítások nyomonkövetése
- ☀ Közvetlenül elérhető metrikák
- ☀ Térben és időben elosztott átvizsgálás

Speciális kollaborációs eszközök

Chat

Overall:

Line 36:

SB: What about n==2 and n==3?
JC: Oh yeah, you're right.
SB: Created Defect D16: Handle cases n==2, 3

Line 41:

SB: Should the round up instead of down?
JC: No, for an upper bound you can round down.
SB: Ok, makes sense.

Code Review Summary:

Review #39: //depot/demo/primes/PrimeUtils.java base version #1 checked in on 2006-12-11 00:00

```

1 // Return the n'th prime number
2 //
3
4 public static int getNthPrime( int n )
5 public static int getPrime( int n )
6
7 {
8
9
10
11 int primeCounter = 0;
12 int k;
13
14 // Scan the primes
15
16 Skipping 15 lines...
17
18 // Return true if <code>n</code> is prime, false if composite
19 //
20
21 public static boolean isPrime( int n )
22 {
23 // Initial check for divisibility by 2 or 3 takes care
24 // of 2/3rds of the cases!
25
26 if ( (n%2) == 0 || (n%3) == 0 )
27 return false;
28
29 // Don't have to test divisors all the way up to `n`.
30 // Biggest possible divisor is sqrt(n).
31 final int maxDivisor = (int) Math.sqrt( n );
32
33 // Scan for non-trivial divisors of n.
34 for( int k = 5 ; k <= maxDivisor ; k++ )
35 for( int k = 2 ; k < n ; k++ )
36 {
37 if ( ( n % k ) == 0 ) // if anything divides us, we're composite
38
39
40
41
42
43
44
45
46
47

```

CodeCollaborator

Speciális kollaborációs eszközök (folyt.)

Results

Recently Completed Reviews

Report created for Jason Cohen on 2007-09-07 10:33
Review Creation Date: between Wed Aug 01 2007 and Sat Sep 08 2007
Defect Count: greater than 1
Sort #1: ID descending

ID	Review Title	Review Creation Date	Author Full Name	Defect Count	LOC Changed	Total Person-Time
1305	Off view improvements	2007-09-05 17:01	Brandon Dufrette	1	35	00:40:58
1302	Support modifying changelist for addactivity and addtrack	2007-09-04 15:18	Eric Brown	6	444	02:11:19
1305	User version editing for addactivity, addtrack	2007-09-04 15:18	Eric Brown	12	705	03:21:00
1307	auto-detect SCM system	2007-09-04 11:03	Roy Patterson	1	2122	01:17:50
1353	Added system option to disable auto-reopen of completed reviews	2007-08-30 22:24	Jason Cohen	4	265	01:41:11
1302	Improve performance of diff widget and	2007-08-22 15:47	Brandon Dufrette	1	133	00:22:34

Review Metrics: A real world case study

Review Metrics: A real world case study

Review Metrics: A real world case study

CodeCollaborator