Szoftverellenőrzési technikák (vimim148)

Automatikus teszt futtatás, folytonos integráció

Ujhelyi Zoltán, Micskei Zoltán,
Monostori Dénes

http://www.inf.mit.bme.hu/

Utolsó módosítás: 2014.11.1.

Az előadás felhasználja Monostori Dénes fóliáit.

Folytonos integráció

*"Continuous Integration is a software development practice where members of a team **integrate their work frequently**, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is **verified by an automated build** (including test) to detect integration errors as quickly as possible."*

Martin Fowler

http://www.martinfowler.com/articles/continuousIntegration.html

2

Folytonos integráció

- Gyakori agilis technika
- Céljai
  - **Minőség** növelése
  - Piacra kerülési idő csökkentése
- Build szerver
  - Automatikus integráció támogatása
- Követelmények
  - Jól definiált fejlesztői (és fordítói környezet)
  - Közös verziókezelő rendszer

3

A folyamatos ellenőrzés miatt csökken a piacra kerülési idő: hamarabb felfedezett hiba olcsóbban javítható

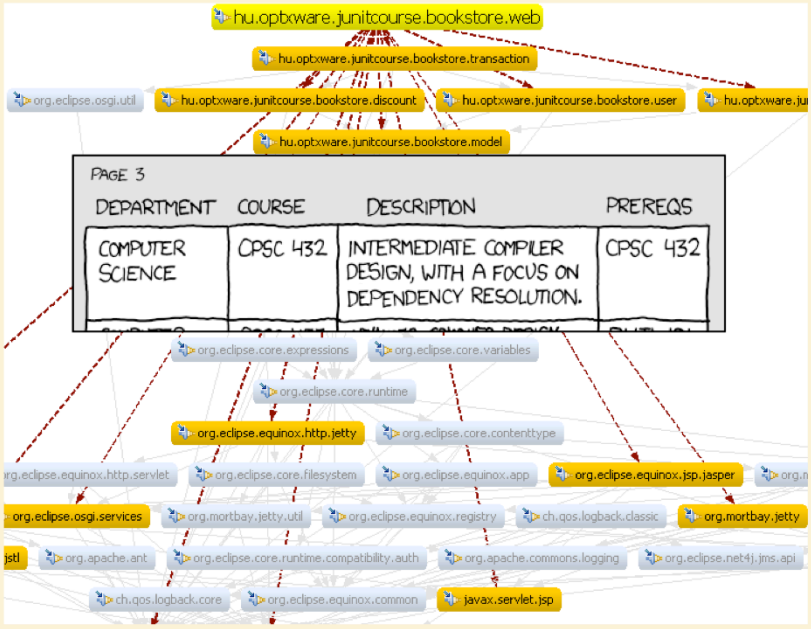
Példa: Eclipse plug-inek fordítása

- Eclipse plug-in
  - Beépülő modul
  - Jól definiált függőségek
  - Verziószámok
- De önmagában nem
  - Fordul
  - Fut
  - Tesztelésnél izoláció problémás lehet!

4

Újonc kérdései egy senior-hoz

Függőségek (Eclipse plug-in)



Forrás: http://xkcd.com/754/

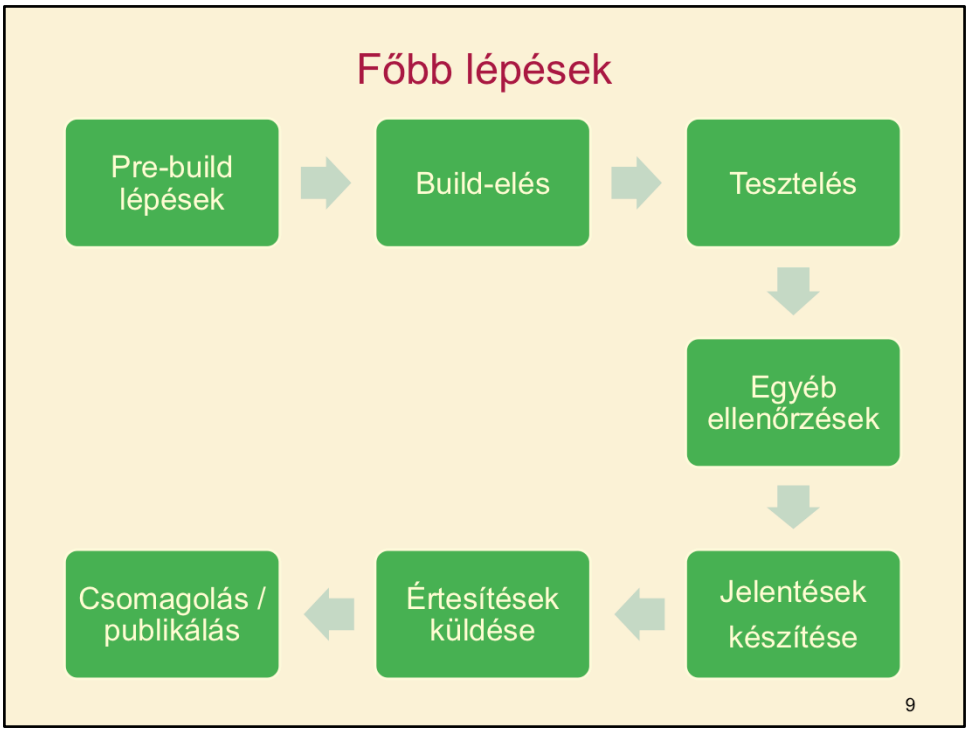
BUILD FOLYAMAT

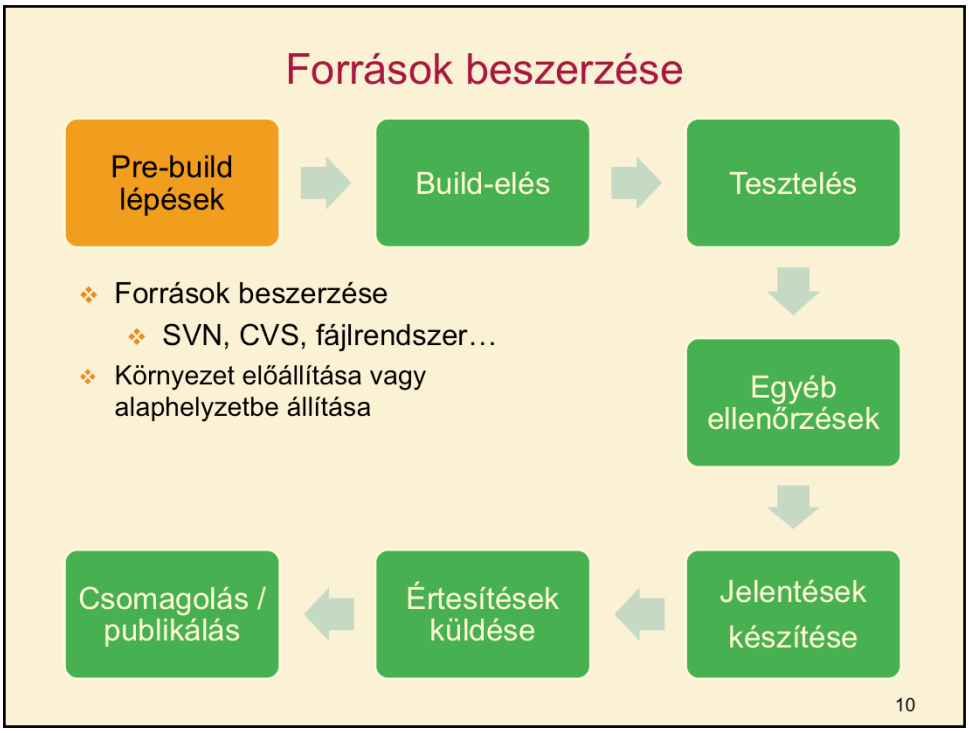
7

Központi build folyamat

- Bonyolult, összetett folyamat
- Cél:
  - Futtatás
    - Egyetlen, **központi helyen**
      - Erőforrásigény
    - **Automatikusan**
      - Nem felejtődik el
  - **Környezetfüggetlen**, fut
    - Fejlesztő gépén parancssorból (GUI nélkül)
    - Fejlesztőkörnyezetben!
    - Build szerveren

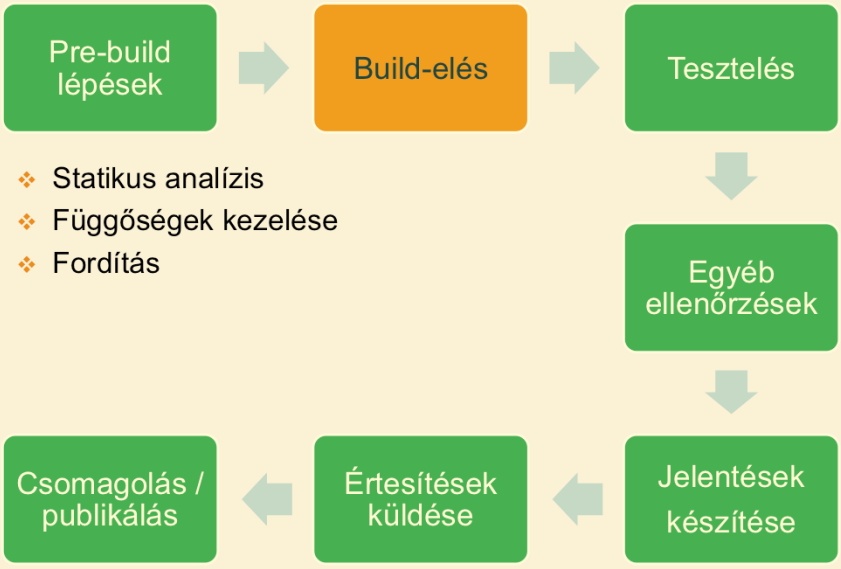
8


Főbb lépések
Pre-build lépések
Build-elés
Tesztelés
Egyéb ellenőrzések
Jelentések készítése
Értesítések küldése
Csomagolás / publikálás
9



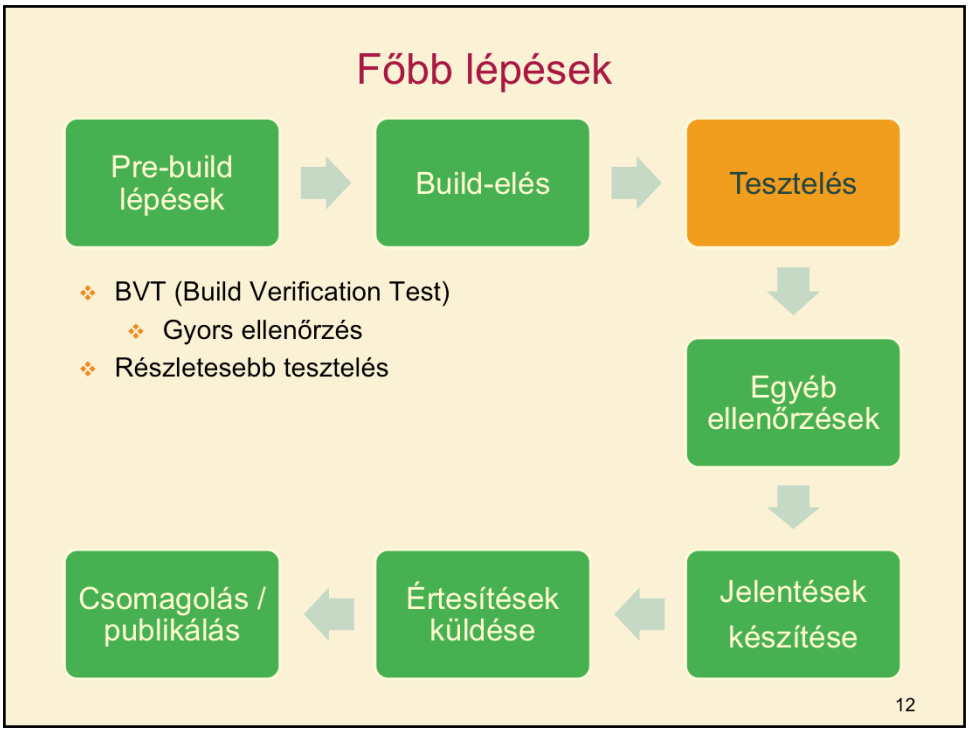
Init pl.: fájlok törölgetése

Főbb lépések



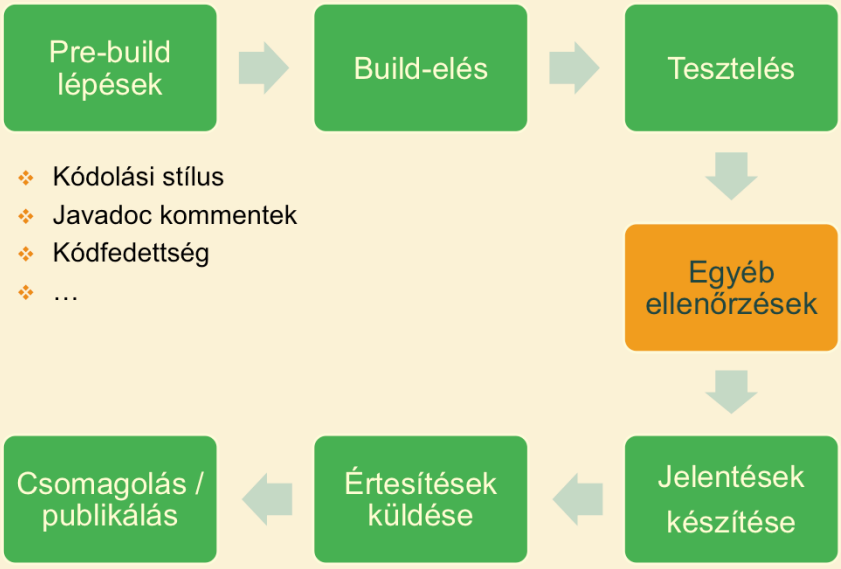
- Statikus analízis
- Függőségek kezelése
- Fordítás

11



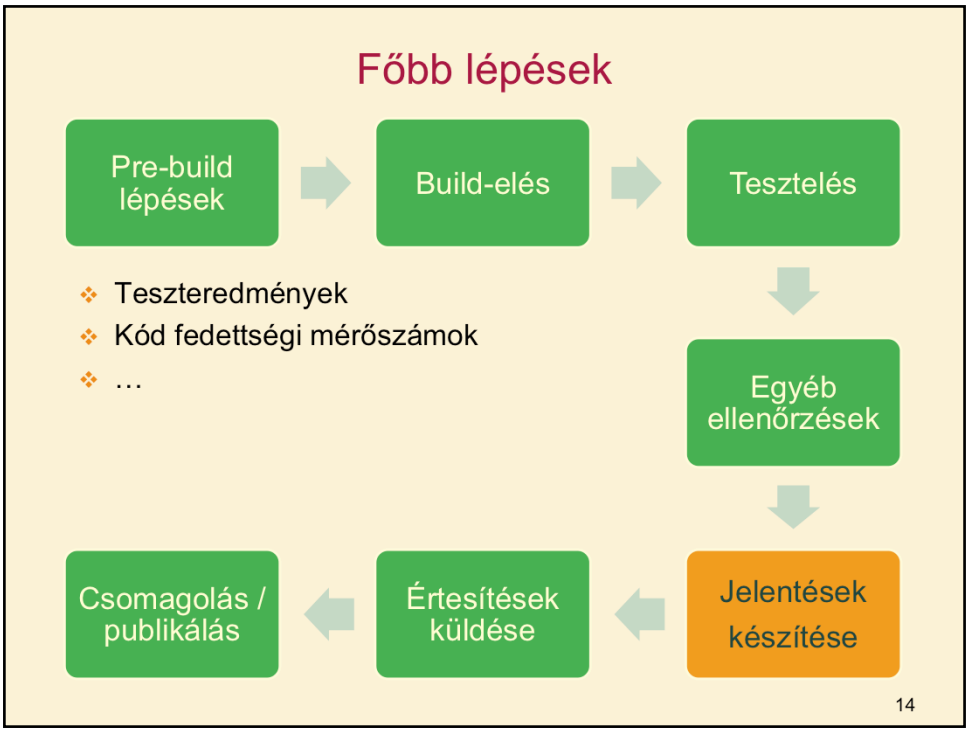
BVT – csak néhány teszt, amik a főbb funkciókat ellenőrzik, így gyorsan eldönthető, hogy érdemes-e tovább tesztelni
Smoke teszt ill. Build Acceptance Test néven is emlegetik

Főbb lépések



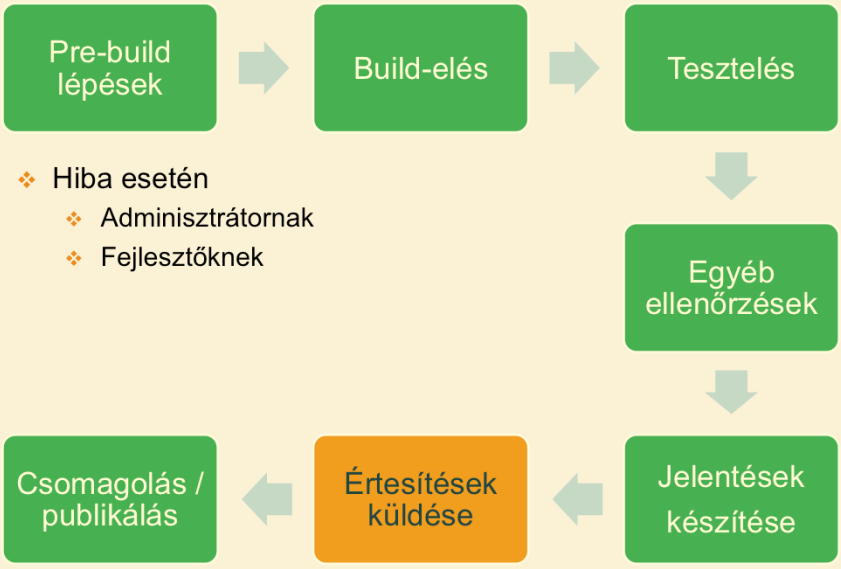
- Kódolási stílus
- Javadoc kommentek
- Kódfedettség
- …

13



Grafikonok, részletes táblázatos eredmény

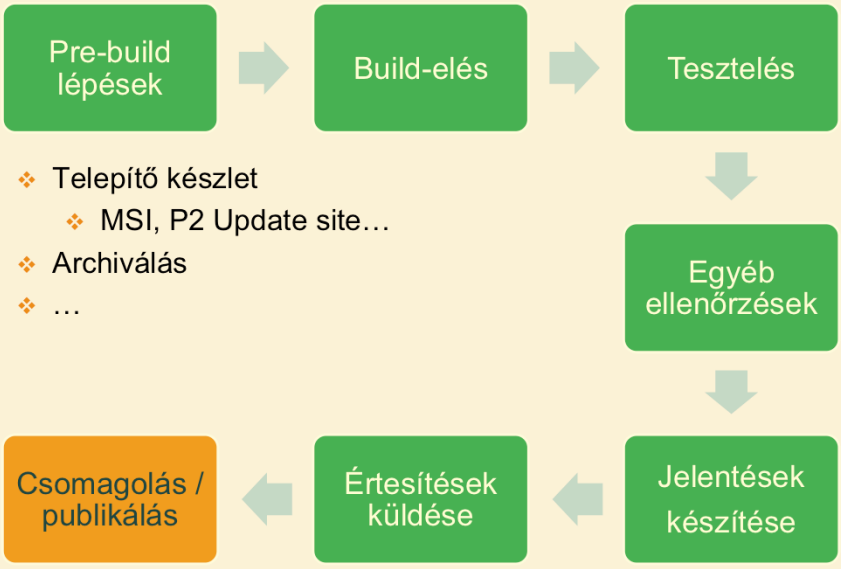
Főbb lépések



- Hiba esetén
  - Adminisztrátornak
  - Fejlesztőknek

15

Főbb lépések



- Telepítő készlet
  - MSI, P2 Update site…
- Archiválás
- …

16

Build idő

- Mit jelent?
  - *“to detect integration errors as quickly as possible”?*

    folytonos integráció definíció
  - Visszajelzés 10-15 percen belül
  - Build folyamat legyen egy óra
  - Minden éjjel fusson le
- Megoldható?
  - Tesztelés ideje?

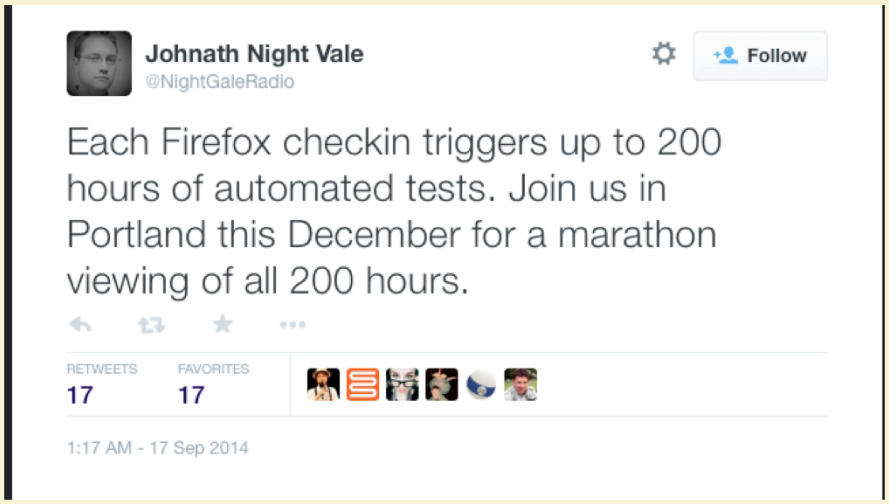
17

Mozilla Firefox (2010)

- 17 platform
- 12 branch forrásnak
- 1200 build and teszt gép
  - Fordítási idő: 12.40 óra
  - Tesztelési idő: 54.48 óra
  - CPU időben: 2.79 nap (!)
  - Korábban release: 10 nap

Forrás: http://relengofthenerds.blogspot.com/2010/11/mozilla-versus-eclipse-build.html

Mozilla Firefox (2014)



Forrás: https://twitter.com/NightGaleRadio/status/512017585672433664

19

Eclipse kiadások

- Build folyamat
  - 9 óra (2010 márciusában)
- Sok fejlesztő
- Sok változat
  - Classic, JEE, Modeling…
- Sokféle platform
  1. Windows, Linux, Mac OSX, Solaris…
  2. Win32, GTK, Motif, Carbon…
  3. x86, x86_64, sparc…
- *“Shipping is hard, that’s why we do it 7 times a release.”*

20

Sok fejlesztő -> másik kódjába bele kell nyúlni -> tesztek kellenek, hogy ezt meg merjék tenni

Visszajelzés gyorsítása

- Első lépés:
  - Profiling
  - Lássuk, hol lassú
- Eredmények alapján
  - Egyszerűsített build
  - Build munkafolyamat

- Eclipse nightly build (2009. 11.)

Forrás beszerzése	20 perc
Digitális aláírás	1 óra 14 perc
Director használata	20 perc
Update site	4 perc
Csomagolása	30 perc
Tesztelés	6 óra 40 perc

Digitális aláírás nem kell mindig

Tesztelés nehezebb ügy

Forrás: https://bugs.eclipse.org/bugs/show_bug.cgi?id=293830#c11

AUTOMATIKUS TESZTFUTTATÁS

22

Automatikus tesztelés

- Teszt futtatás, kiértékelés automatizálása
  - Emberi kiértékelés lassú
  - Hibakezelés
- Manuális vagy automatikus?
  - Nehézség
    - Pl. GUI, CD írás, rajzolás
  - Tesztelés élethossza
    - Meddig kell a teszt, milyen gyakran
  - Pontosság
    - Hibás pozitív (false positive)

23

Automatikus tesztelés tipikus lépései

Setup	• Legfrissebb verzió fordítása/telepítése • Különböző platform, OS, böngésző… • Virtuális gépek, „Lab manager” programok
Execution	• Egyszerű script / xUnit / keretrendszer • Naplózás
Analysis	• Teszt kiértékelése • Sokszor nem triviális
Reporting	• Tesztek ezrei esetén nem elegek a naplófájlok • Összesítő információk
Cleanup	• Ismert, tiszta állapotba visszaállítás • Cél: tesztek ne befolyásolják egymás futását
Help	• Teszt kód is ugyanolyan kód, azt is dokumentálni kell • Sokszor a teszt kód hosszabb, mint az éles

24

Naplózás

- Források változásai
- Figyelmeztetések
- Tesztek
  - ID, név
  - Környezet
  - SUT (System Under Test) információk
    - Verzió
    - Beállítások
    - Nyelv
    - …
  - Eredmények
- …

25

Környezet: OS, architektúra, hardver, nyelv…

Tesztfuttatási stratégiák

- Teljes tesztelés
  - Minden tesztet futtatunk
  - Minden kiadás előtt **kell**
- Smoke tesztelés
  - Megfelelően válogatott teszthalmaz készítése
  - Gyengébb pontosság
    - Lényegében alap funcionalitás (“elindul-e”)
  - Gyorsabb futás
    - Így lehet gyorsan visszajelezni
    - Unit tesztek tipikusan belefférnek

26

Teljes build munkafolyamat

- Több lépcsős ellenőrzés
  - Első lépés
    - Fordítás + smoke teszt
  - Későbbi lépések
    - Integrációs tesztek
    - Teljesítmény tesztek
    - Statikus analízis
    - Manuális tesztelés
- Későbbi fázis tovább tart
  - De lehet ritkábban is futtatni

27

EMF-IncQuery munkafolyamat

28

Hiba azonosítás

- Integrációs teszt
  - Több commit hatását teszteli egyszerre
  - Melyik okozta a hibát?
- Delta debugging technika
  - Azonosítsuk a hibás teszteket
  - Futtassuk őket újra a köztes committokon
    - Rendezett lista -> bináris keresés működik
  - Git verziókezelő beépítve támogatja
    - git bisect parancs

29

Teszt futtatás – nagy léptékben

- Nagy projektek esetén (OS, böngésző, IDE…)
- Tesztek futtatása 10-100-1000 gépen
  - Jó tesztek függetlenek
  - Tesztkészlet jól párhuzamosítható
- Tipikus felállás
  - Teszt vezérlő
  - Teszt adattár (kód, log, jelentések)
  - Ágens a tesztelő gépeken
- Pl.: Rational, Visual Studio, saját megoldások

30

BUILD VÉGREHAJTÓ MOTOROK

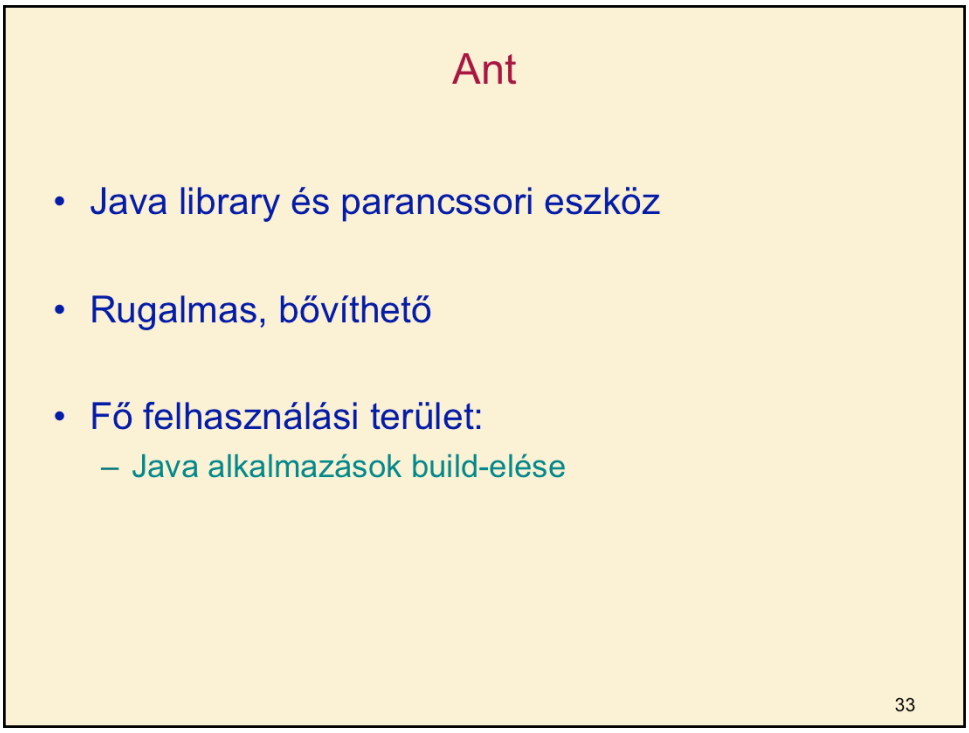
31

Build végrehajtó motorok

- Make
  - C/C++
- Apache Ant
  - Make fájl Java-hoz, XML alapokon
- Apache Maven
  - Egységes forrás letöltés és fordítás
  - Funkcionalitásában hasonlít az Ant-hoz
- …

32

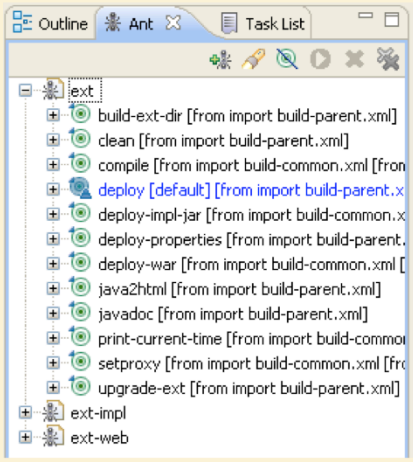
Maven: Java-s projektek menedzselése, build-elés automatizálása



Lényegében bármilyen folyamat megvalósítható vele, ha az leírható célok (target-ek) és feladatok (task-ok) használatával.

Ant alapfogalmak

- Project
  - Build fájlonként egy
- Target
  - Végrehajtandó taszkok egy halmaza
  - 1..*
  - **Egymástól függhetnek**
  - Pl. compile, deploy
- Task
  - Végrehajtható kód
  - Pl. javac, copy, junit, exec, signjar, mail…



34

J2EE alkalmazás fejlesztés – deploy pl. 1 percig tartott
Sokféle beépített taszk, de sajátot is definiálhatunk

További elemek

- Név—érték párok (properties)

```
<property name="build" location="build"/>

<target name="init">
    <mkdir dir="${build}"/>
</target>
```

- Útvonalak, classpath

```
<classpath>
   <pathelement path="${classpath}"/>
   <pathelement location="lib/helper.jar"/>
</classpath>
```

- Bármely projektelemnek lehet ID-ja
  - → Minden hivatkozható

35

Előkészületek

- Szükséges:
  - junit.jar
  - ant-junit.jar
    - Alapértelmezett helye: `ANT_HOME/lib`
- junit.jar megadása:
  - `ANT_HOME/lib` könyvtárba másolással, vagy
  - `-lib` argumentummal, vagy
  - `<junit>` taszk `<classpath>` elemében

36

Példa

```
<project default="test" >
    <path id="classpath.test">
        <pathelement location=„lib/junit.jar" />
        <pathelement location="${build}" />
    </path>

...

    <target name="compile-test">
        <javac srcdir="${tst-dir}" >
            <classpath refid="classpath.test"/>
        </javac>
    </target>
...
```

Példa (folytatás)

```
...
<target name="test" depends="compile-test" >
    <junit printsummary="yes" haltonfailure="yes">
        <classpath refid="classpath.test" />
        <formatter type="plain" />
        <test name="hu.optxware.junitcourse.
            bookstore.book.test.BMListTest"
              haltonfailure="no" outfile="result" >
            <formattertype="xml"/>
        </test>
    </junit>
</target>
```

38

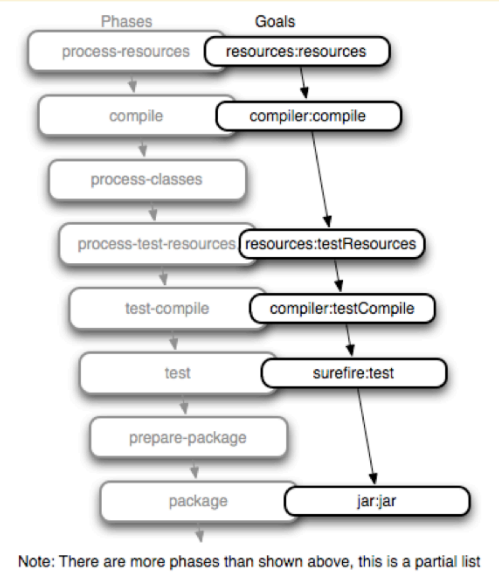
http://ant.apache.org/manual/index.html oldalon megtalálható minden beépített taszkhoz annak részletes leírása (attribútumok, beágyazott elemek, példák)

Maven

- Leíró
  - pom.xml: projekt modell
  - Archetípus: minta
  - Eltérések felsorolása a mintától
- Fordítás
  - Megnevezünk egy célt (pl. teszt, csomagolás)
  - Végignézi az összes szükséges fázist
- Magas szintű fogalomkészlet
  - Függőségkezelés
  - Repository, mirror repository
  - …

39

Maven életciklus és célok



Példa: JUnit teszt futtatás Mavennel

Projekt struktúra

- my-app
  - pom.xml
  - src
    - main
      - java
        - » com |
          - » mycompany
            - » app |
              - » App.java
  - test
    - java
      - com
        - » mycompany
          - » app
            - » AppTest.java

41

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://
www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://
maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>Maven Quick Start Archetype</name>
  <url>http://maven.apache.org</url>
  <packaging>jar</packaging>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.0</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Metaadatok

Java alkalmazás

JUnit függőség

Mi hiányzik?

42

Példa: JUnit teszt futtatás Mavennel

- Projekt struktúra
  - Alapértelmezett van minden alkalmazástípushoz
  - Felülbírálható
  - Ugyanakkor Maven plug-inek olvassák!

Ant vs Maven

- Igazi “vallásháború”
- Ant
  - Minden kézben tartható
  - Egyedi projektnél hasznos
- Maven
  - “Convention over configuration”
  - Minden Maven projekt hasonló…
  - Függőségkezelés

44

Egyéb eszközök

- Gradle
  - Groovy scriptekkel leírt build folyamat
  - Maven repository-t érti
- MSBuild
  - Microsoft build technológia
- Rake
  - Ruby alapú keretrendszer
- …

45

BUILD SZERVER ESZKÖZÖK

Eszközök

- Apache Continuum (Java)
  - XML szerkesztés + webes UI
- CruiseControl (Java, .NET, Ruby)
  - XML szerkesztés
- Hudson/Jenkins (Java, de kiterjeszthető)
  - Webes UI
- TeamCity (Java, .NET, Ruby)
  - Fizetős
- Travis CI
  - Egyszer használatos virtuális gép a buildre
  - GitHub integráció
- …

47

A Hudson UI-e az egyik legjobb az open-source build szerverek közül.

Hudson (Jenkins)

- Java szervlet alapú
  - Tetszőleges alkalmazás szerveren fut
- Plug-in alapú, **bővíthető**
- Frissítések keresése automatikus
- Gyorsan bele lehet tanulni
- Nem végez tényleges fordítást
  - Időzítés
  - Menedzselés
- Több folyamat, köztük akár **függőségekkel**

48

Alkalmazás szerverek: Glassfish, Jboss, WebSphere Application Server …

Épp Xtext és EMF build megy, de van itt mindenféle Eclipse-es projekt
Abból is látszik, hogy jó a Hudson, hogy az Eclipse háza táján is ezt használják
Kék: sikeres, Sárga: instabil, Piros: hibás build
Felhőcskék: utóbbi időkben mennyire voltak sikeresek a build-elések

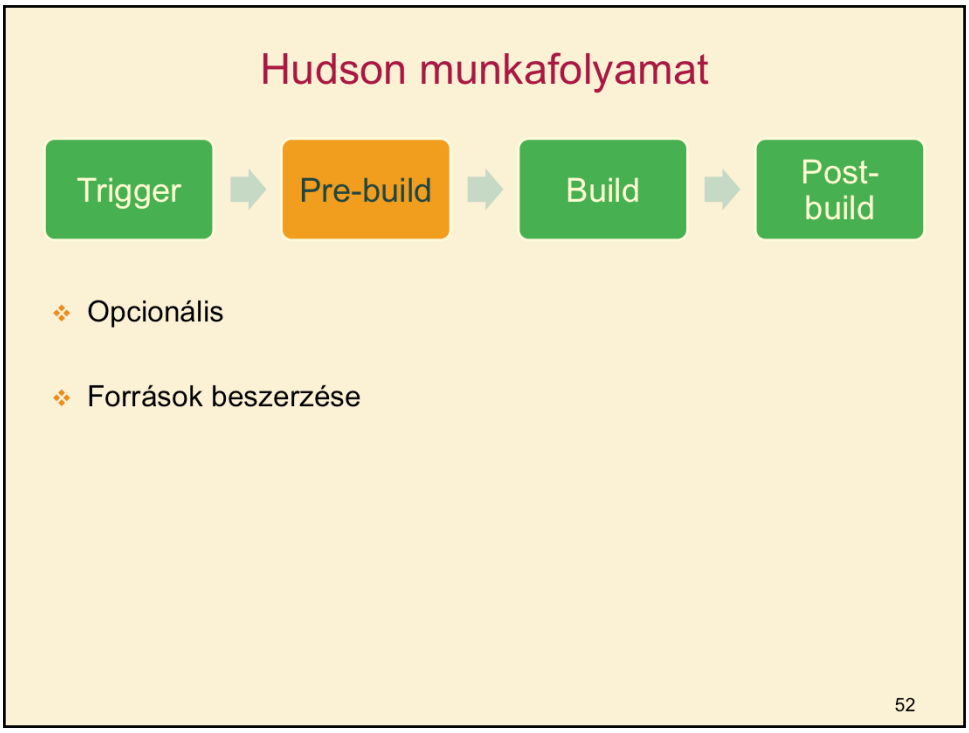
Hudson munkafolyamat

50

Hudson munkafolyamat

- Kézi
- Időzített
- Verziókezelő rendszer változása
- Függő job befejeződése
- Egyéb (bővíthető)

51



Verziókezelő rendszer változásait is feljegyzi

Hudson munkafolyamat

- Tényleges fordítási lépések
- Végrehajtó támogatás
  - Ant
  - Shell script
  - Maven
  - Bővítményekkel kb. bármi

53

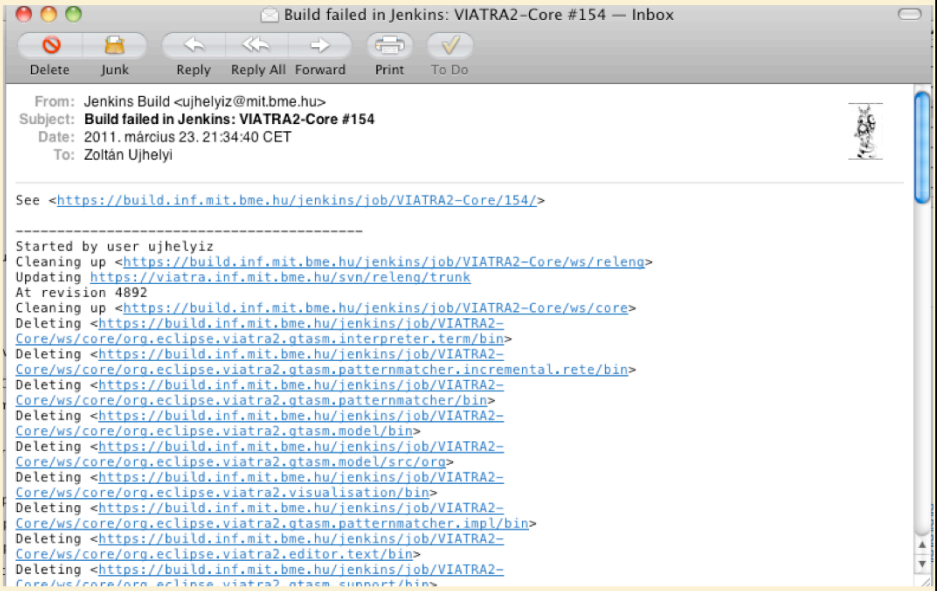
Hudson munkafolyamat

- Opcionális

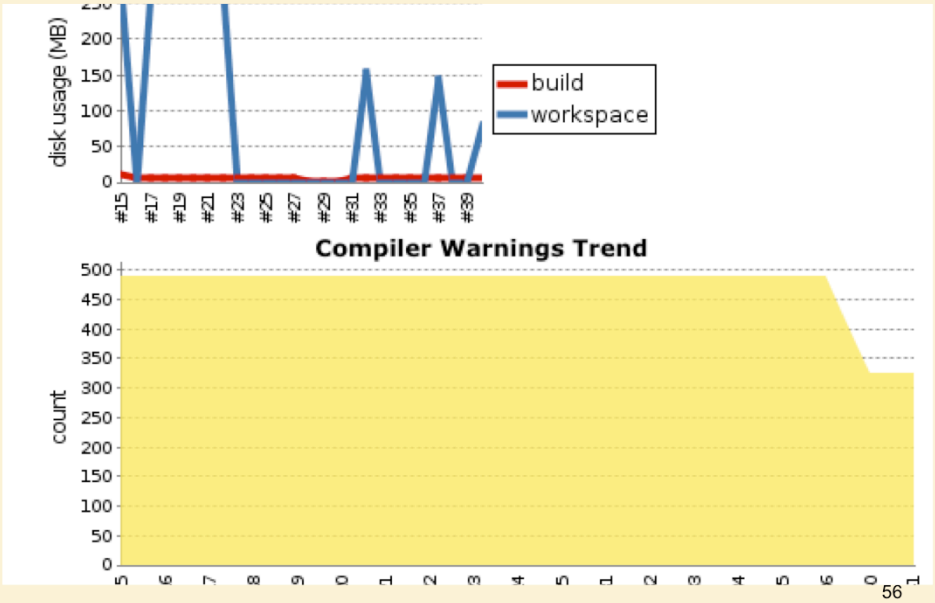
- Archiválás
- Publikálás
- Függő build-ek indítása
- Értesítések
- …

54

Blame mail

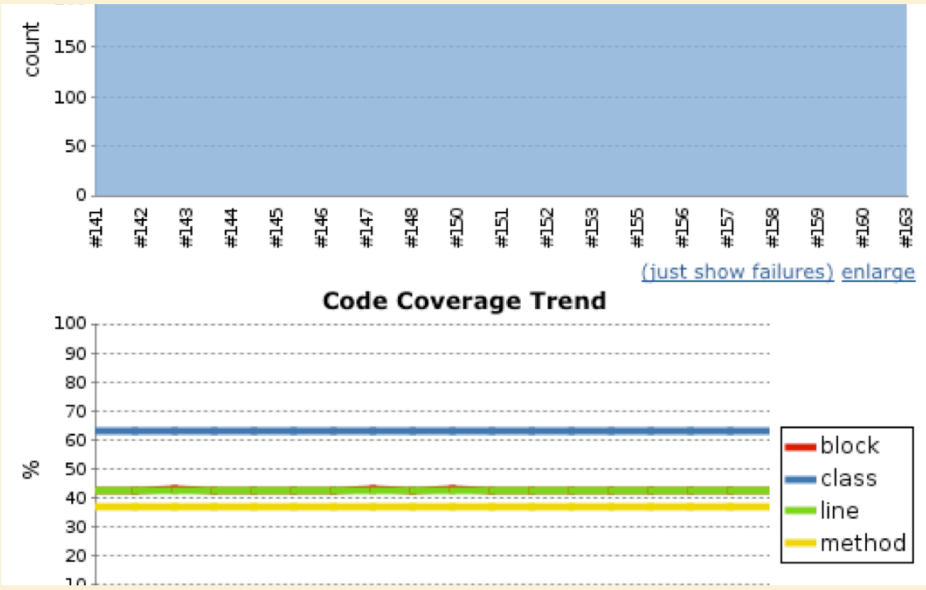

Build failed in Jenkins: VIATRA2-Core #154 — Inbox
Delete
Junk
Reply
Reply All
Forward
Print
To Do
From: Jenkins Build <ujhelyiz@mit.bme.hu>
Subject: Build failed in Jenkins: VIATRA2-Core #154
Date: 2011. március 23. 21:34:40 CET
To: Zoltán Ujhelyi
See <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/154/>
Started by user ujhelyiz
Cleaning up <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/releng>
Updating https://viatra.inf.mit.bme.hu/svn/releng/trunk
At revision 4892
Cleaning up <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/core>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/core/org.eclipse.viatra2.gtasm.interpreter.term/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/core/org.eclipse.viatra2.gtasm.patternmatcher.incremental.rete/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/core/org.eclipse.viatra2.gtasm.patternmatcher/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/core/org.eclipse.viatra2.gtasm.model/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/core/org.eclipse.viatra2.gtasm.model/src/org>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/core/org.eclipse.viatra2.visualisation/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/core/org.eclipse.viatra2.gtasm.patternmatcher.impl/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-Core/ws/core/org.eclipse.viatra2.editor.text/bin>
Deleting <https://build.inf.mit.bme.hu/jenkins/job/VIATRA2-

Metrikák, trendek



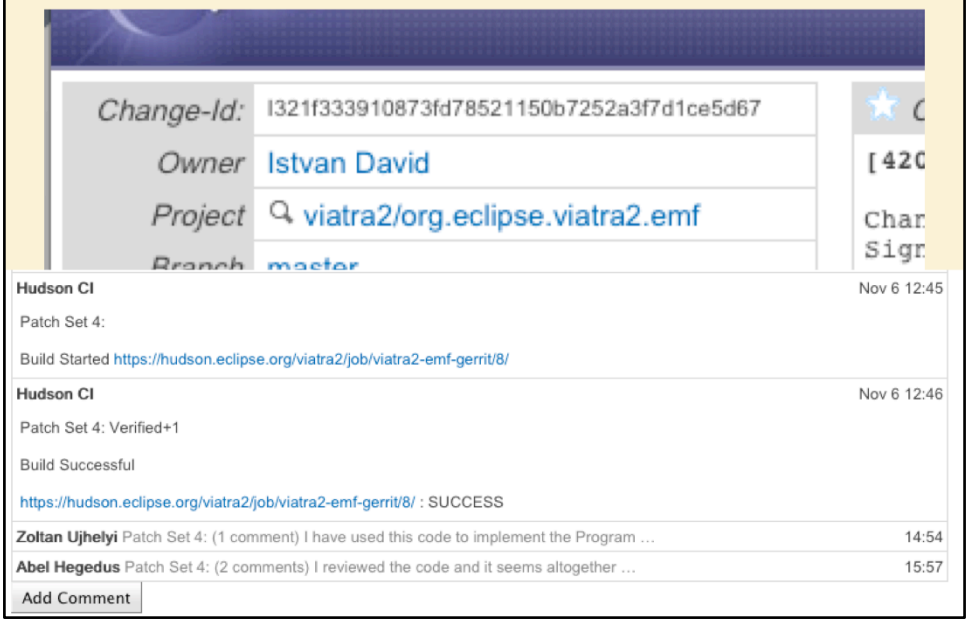
56

Kódfedettség trendek

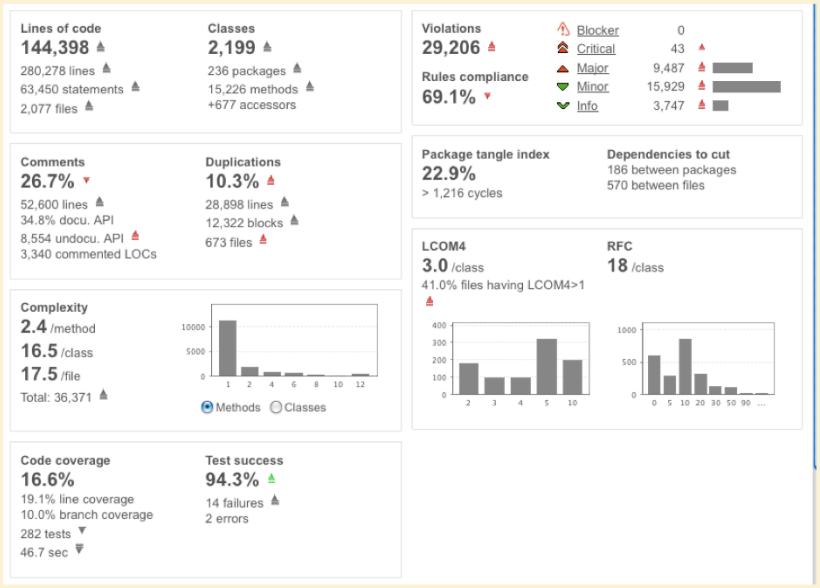


57

Code Review Integráció (Gerrit)



Egyéb metrikák (Sonar)



További lehetőségek

- Build slave-ek
  - További Hudson példányok kezelése
- Munkafolyamatdefiníció
  - Több job egymás után
- Sokféle bővítmény
  - Trigger
  - Jelentések
  - Közzététel

60

GYAKORLATI TIPPEK

61

Verziókezelő rendszer

- Minden legyen verziókezelőben
  - Kód
  - Build scriptek
    - Függőségdefiníció
  - Adatbázisséma
  - …
- Ökölszabály
  - Új fejlesztő
  - Új gép
  - Checkout + build működik

62

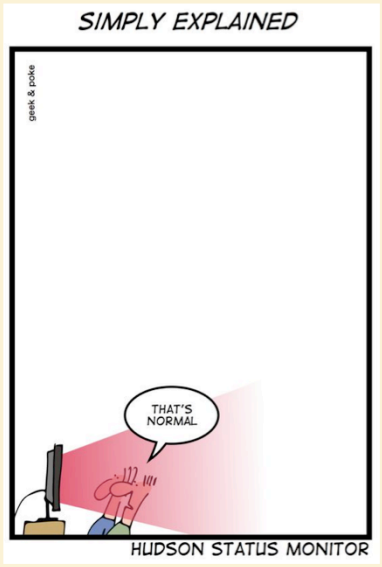
Verziókezelő rendszer

- Commit early, commit often
  - Praktikusan **minimum** napi 1, de inkább több commit
  - Minden commit egy funkcionális egység
  - Commit komment!
- Hosszú távú kísérleti ágak
  - Branchek a verziókezelőben
    - Folyamatosan frissíteni fejlesztői ággal
  - Ugyanúgy tesztelhető, mint a normál ág

63

Hibák kezelése

- Build job nem működik
  - Azonnal javítani
  - Nincs több módosítás
- Cél
  - Release bármikor lehet



ÖSSZEFOGLALÁS

65

Teszt automatizálás

- Bonyolult problémák
  - Összetett folyamat
  - Sok részlépés
- Külön-külön is automatizálható
  - Fokozatos bevezethetőség
- Fejlesztői és üzemeltetői ismereteket is igényel

66

Összefoglalás

- Folytonos integráció
  - Nem technológia
  - Fejlesztési módszertan
- Build folyamat
  - Minimálisnál nagyobb projekt esetén „kötelező”
  - Jó eszköztámogatás
  - Nem triviális beállítani

67