Nástroje pro skupinovou spolupráci

Ing. Petr Ferschmann

SoftEU s.r.o.

Ing. Jindřich Vimr

HSF s.r.o.

Překladač nestačí

Co potřebujete, abyste vytvořili program?

- překladač
- IDE nebo textový editor
- debugger

Je to všechno?

Problémy

- Různé platformy.
- Vydání opravy rok staré verze.
- Obsahovala starší verze nějakou chybu?
- Je vydaná verze stejná jako testovaná?
- Commit před ochodem domů.
- Ztracené papírky.
- Oprava chyby způsobí jinou.
- Roztříšťenost koncentrace.

Co potřebujete?

- **buildovací systém** (Ant, Make, Maven, ...)
- **verzovací systém** (CVS, SVN, BitKeeper, ...)
- **bug tracking systém** (Bugzilla, RT, ...)
- **kontinuální kompilace**
- **automatizované testování** (JUnit, CPPUnit)
- **systémy sdílení znalostí** (Wiki, nástěnka, Google, ...)
- **statické analyzátory kódu** (Checkstyle, ...)
- **profilovací nástroje**
- **generování a správa dokumentace**

- **velký hrnek na čaj, kafe; pizzu**
- **sluchátka**

Verzovací systémy

- Úložiště všech změn v projektu
- Archiv starších verzí
- Nástroj pro paralelní vývoj

Verzovací systémy souvisí s SCM - Software Configuration Management

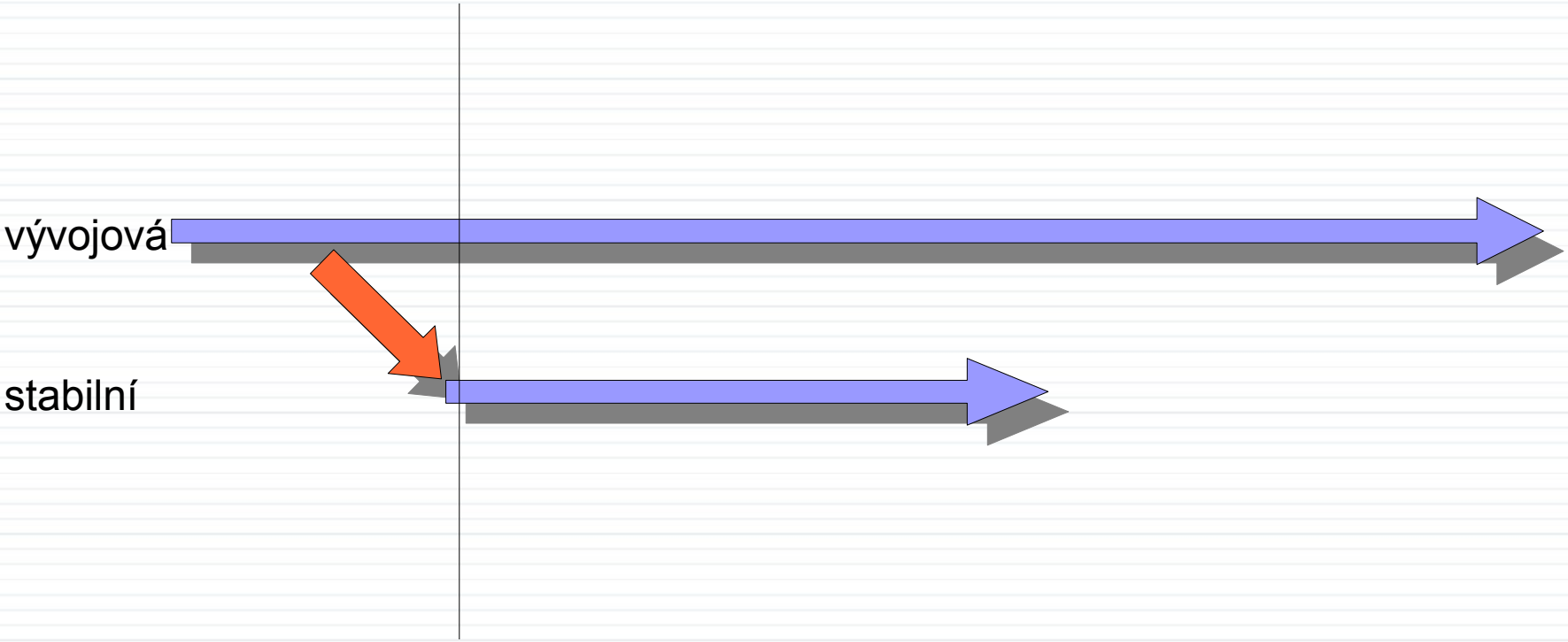
- SCM není jen verzovací systém
  - určuje konfiguraci různých variant systému
  - dnes obvykle podmíněným překladem

Verze produktu

- Firma může mít několik verzí programu
  - **vývojová**
  - **zmrazená vývojová** pro vydání
  - **experimentální** pro konkrétní vlastnost
  - **stabilní** – jen opravy chyb
  - **předchozí stabilní** – ale stále podporovaná
  - …
- To jsou větve (branch) nebo značky (tagy) ve verzovacím systému

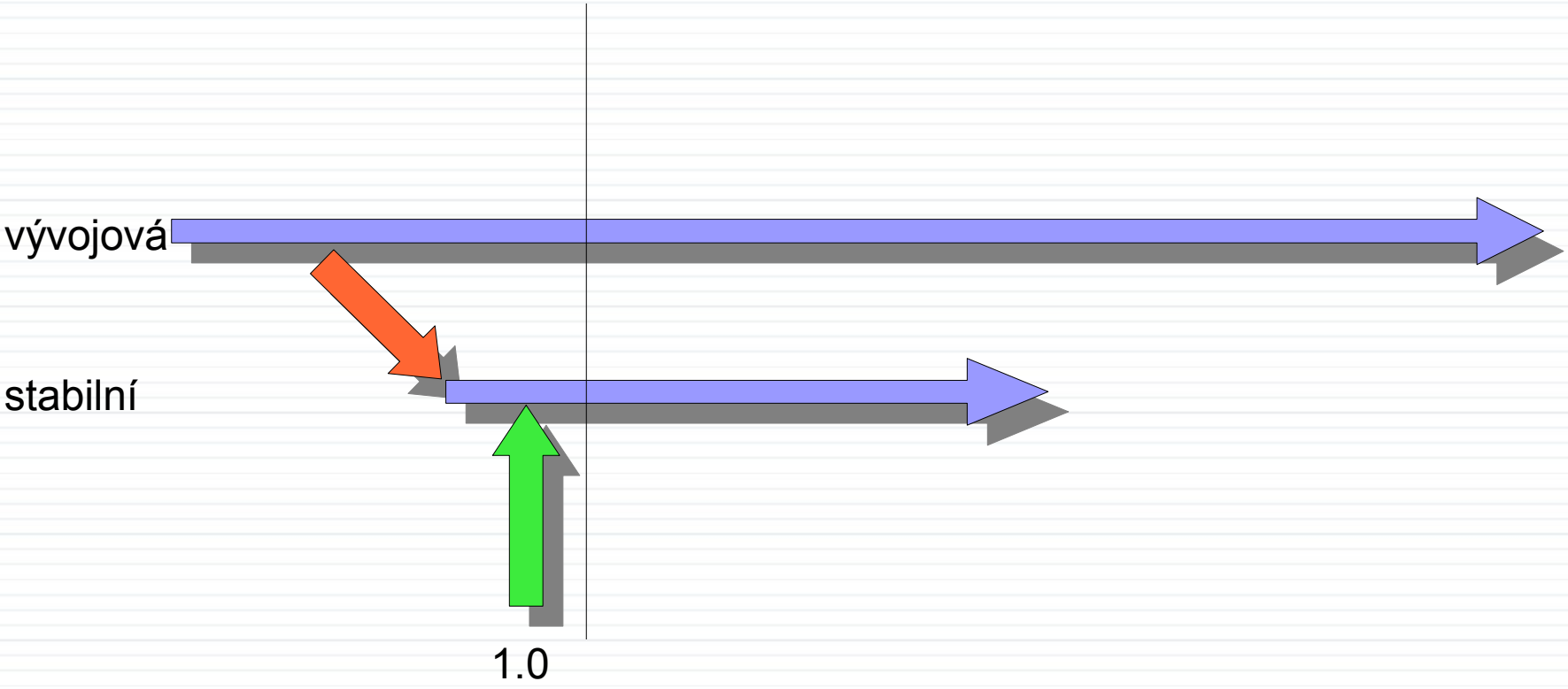
- Vždy různé zdrojáky, ale změny se předávají mezi větvemi
  - » merge

Ukázka větvení



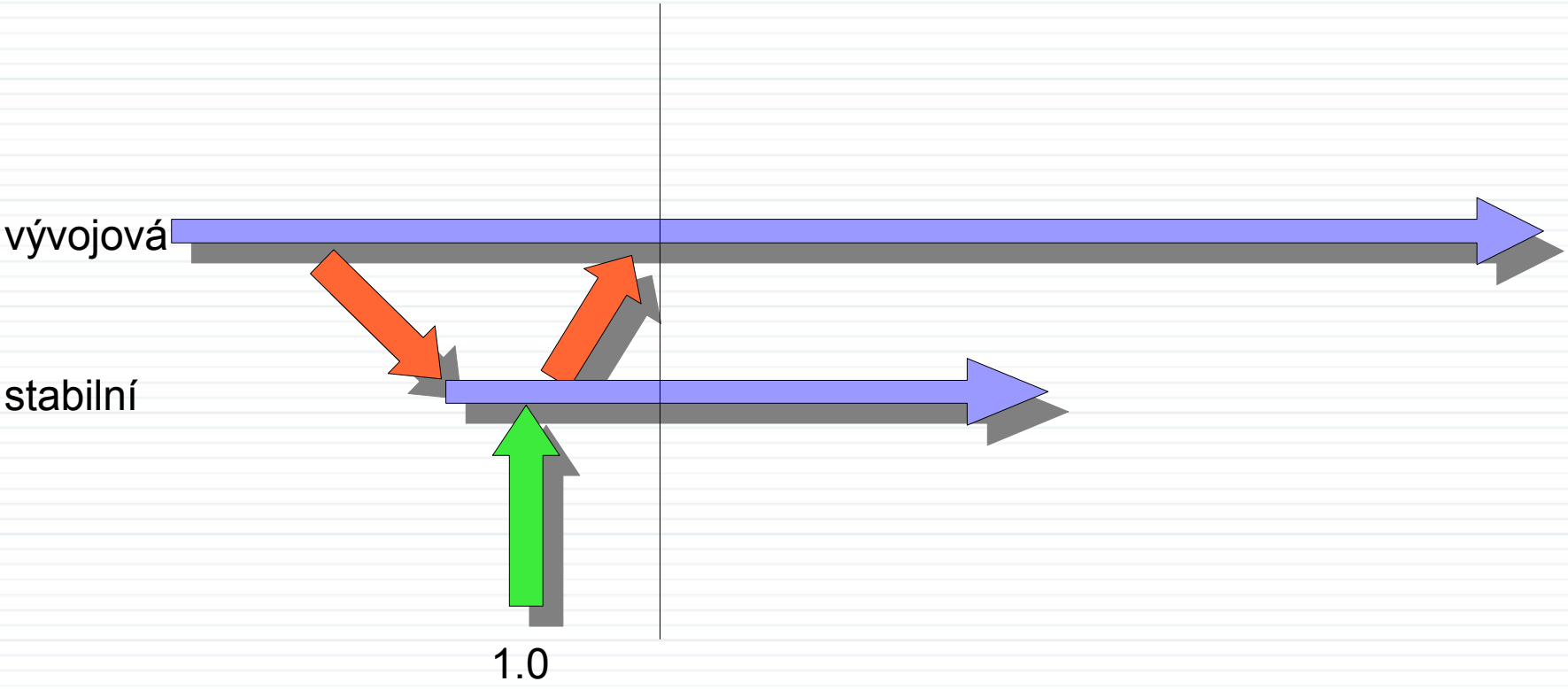
větev merge/split tag

Ukázka větvení



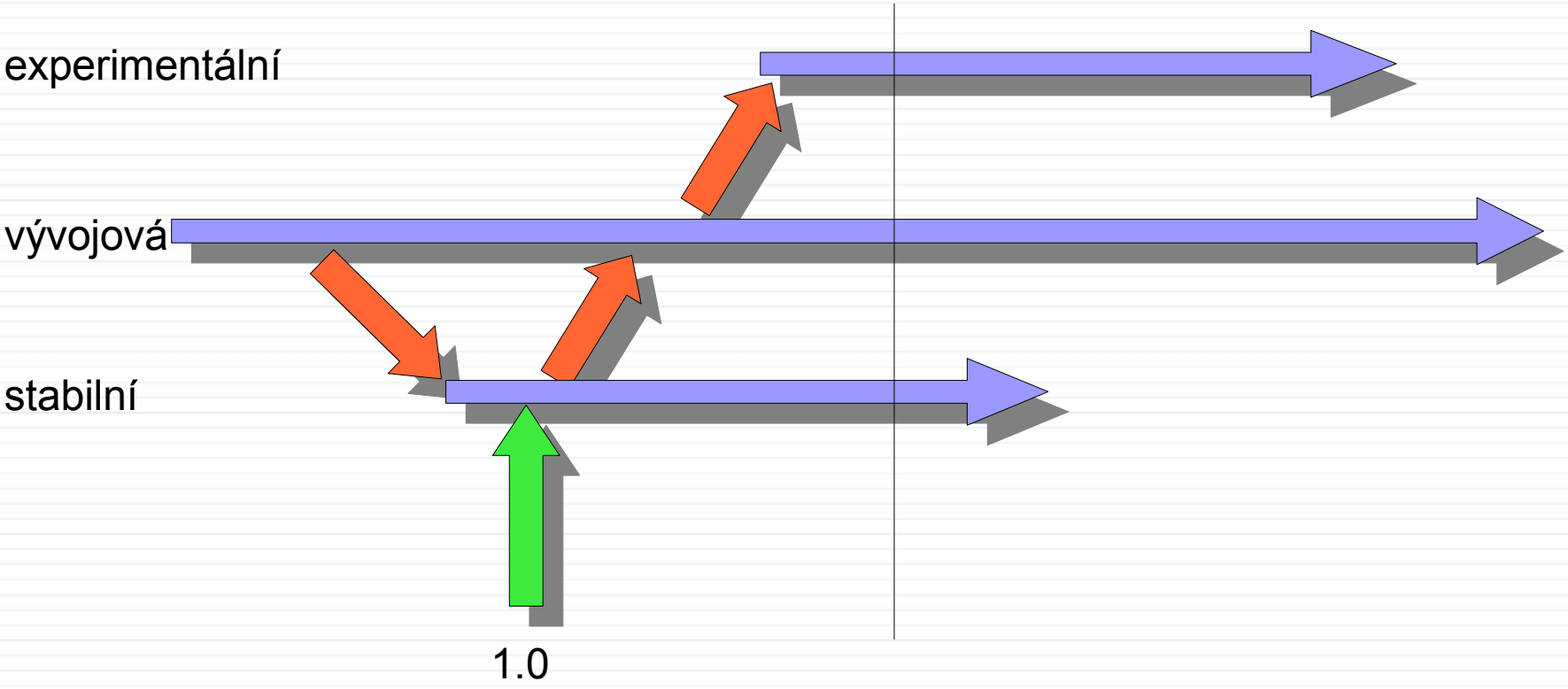
větev merge/split tag

Ukázka větvení



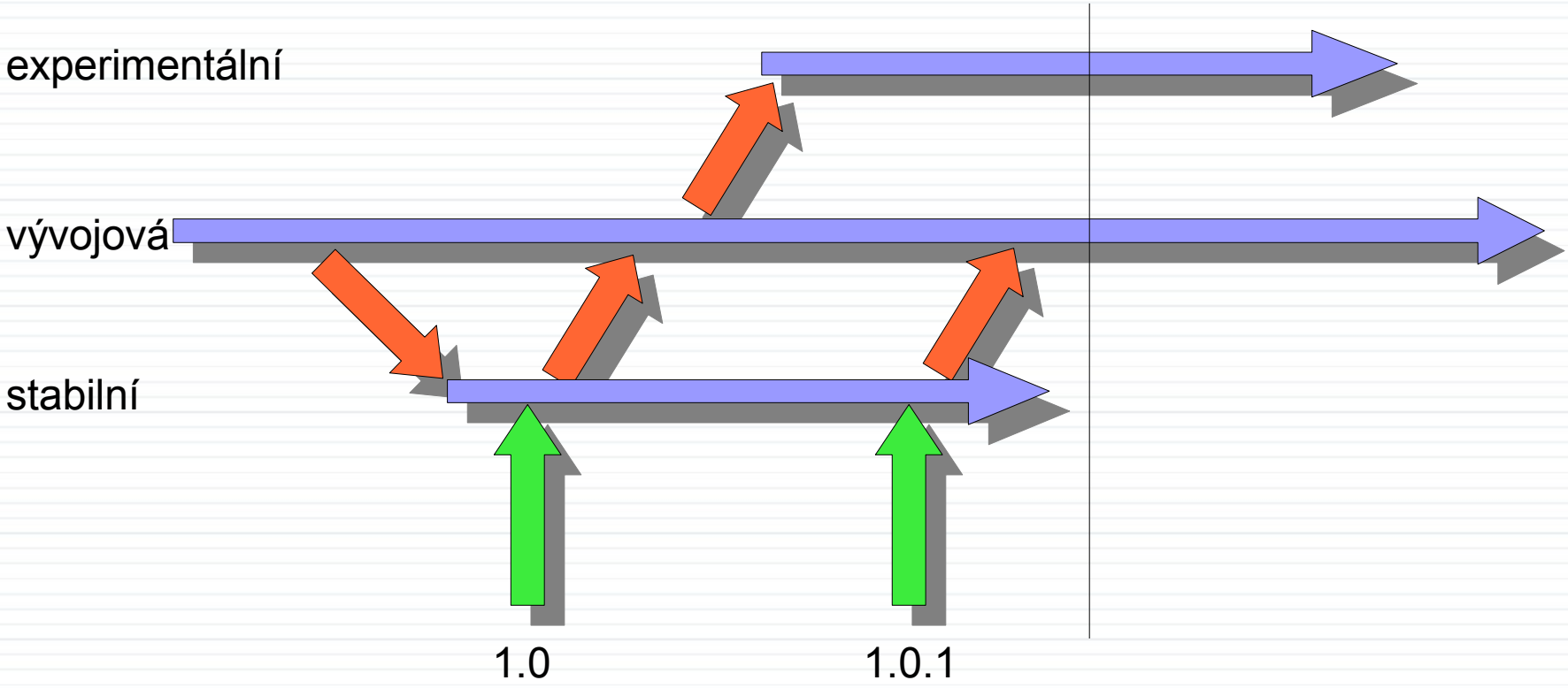
větev merge/split tag

Ukázka větvení



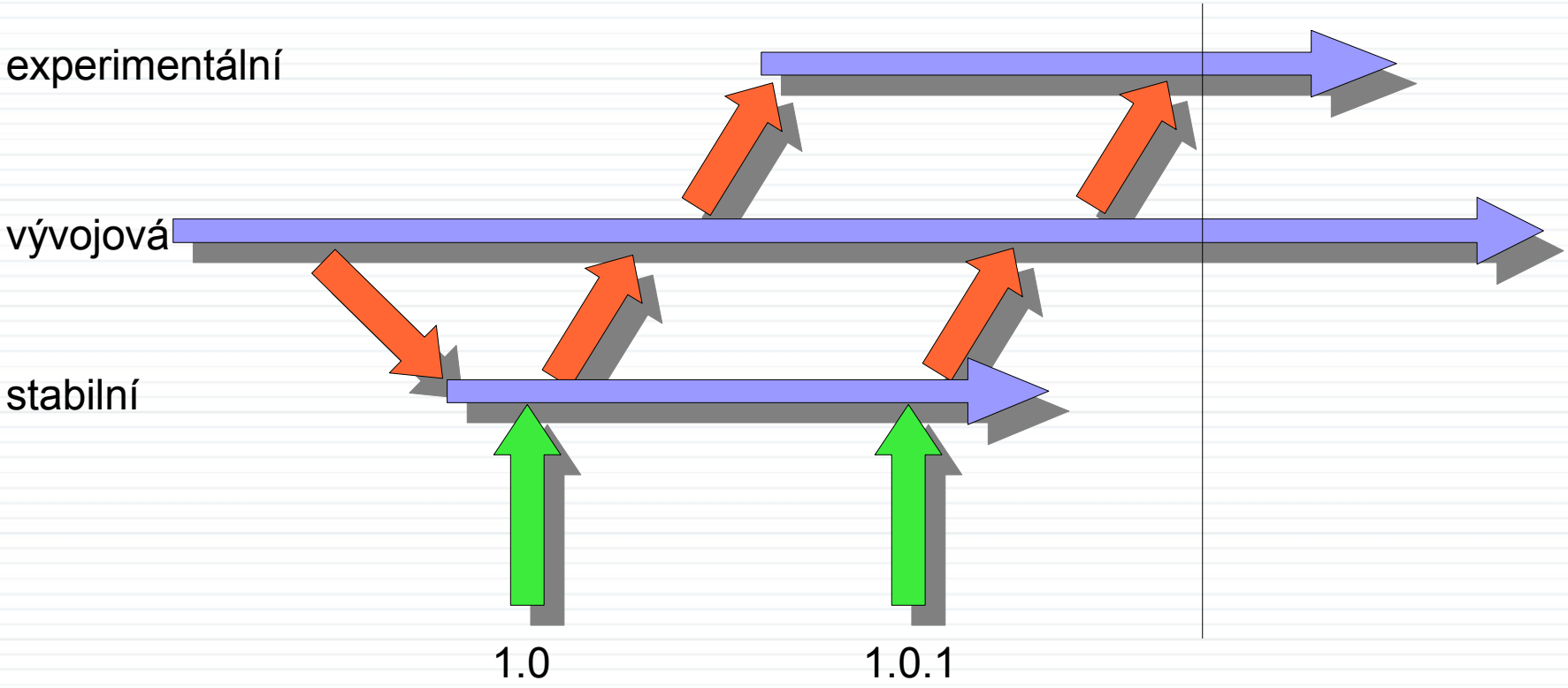
větev merge/split tag

Ukázka větvení



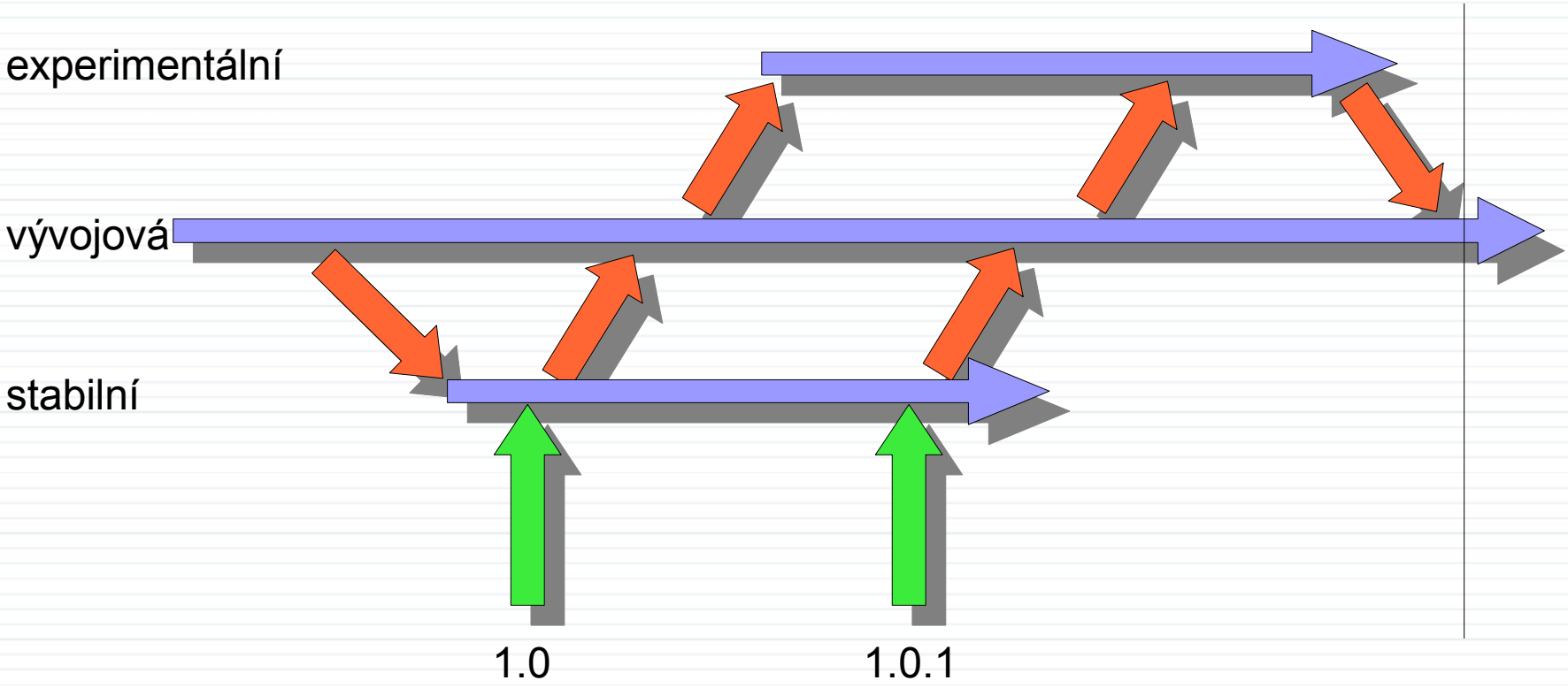
větev merge/split tag

Ukázka větvení



větev merge/split tag

Ukázka větvení



větev merge/split tag

Nejznámější programy

- OpenSource
  - Subversion (SVN)
  - CVS (je nahrazováno SVN)
  - GNU Arch
  - GIT
  - …
- Komerční
  - BitKeeper
  - SourceSafe
  - Perforce
  - ClearCase
  - ...

http://en.wikipedia.org/wiki/List_of_revision_control_software

http://en.wikipedia.org/wiki/Version_control_system

http://en.wikipedia.org/wiki/Software_Configuration_Management

Přístup přes web

exekutori

Revision Information	
Current Directory:	/evidence/
Rev:	2808
Author:	aha
Last modification:	Rev 2808 - 2005-10-07 17:27:25 +0200 (Fri, 07 Oct 2005)
Log message:	B#1825 - pridani sazeb na editaci nakladu
	(Hide changed files)
Modified files:	evidence/trunk/templates/css/pes_base.css.ftl evidence/trunk/templates/fw/spis/i-naklady-obsah.ftl evidence/trunk/templates/fw/spis/i-naklady.ftl evidence/trunk/templates/fw/spis/spis-naklady/action-edit.ftl

[/] [**evidence/**] - View Log - Compare with Previous - Tarball - XML

Path	Log	Tarball	RSS feed
evidence/	View Log	Tarball	XML
branches/	View Log	Tarball	XML
tags/	View Log	Tarball	XML
trunk/	View Log	Tarball	XML

Compare Paths

Sledování změn

- Narazíte na kód, který se zdá nelogický (chybný)
  - nejste si jistí, zda jej můžete smazat/opravit
  - musíme zkontrolovat, proč byla změna provedena

- Musíme tedy zjistit
  - **proč** byla změna provedena
  - které **další změny** s ní souvisí
  - jak zopakovat chybu, kterou změna opravuje

- **Propojení BugTracking systému s verzovacím**

  Přidání čísla bugu ke komentáři u změny

  např.: `Pridan test, ktery take selhava - #1762`

Propojení chyby k bugu

- Je možné sledovat změny k danému bugu

------- *Additional Comment #1 From Petr Ferschmann 2005-09-29 12:01* [reply] ----

```
[svn] commit - exekutori (rev 2744)
Pridan test, ktery take selhava - #1762

[files]
U   evidence/branches/platby/test/cz/softeu/evidence/test/PlatbyTest.java

[web]  https://www.in.softeu.cz/tools/repo/?repo=exekutori&rev=2744
[diff] https://www.in.softeu.cz/tools/diff/?repo=exekutori&rev=2744
```

Obviňování (blame)

- Webové prohlížeče repository umožňují zjistit kdo, kdy provedl změnu

```
14   1019 fers
15   475  fers    import cz.softeu.fw.FWException;
16   1181 fers    import cz.softeu.fw.auditlog.Historizable;
17   1181 fers    import cz.softeu.tools.component.HibernateManager;
18   475  fers
19   223  stankar /**
20   223  stankar  * @hibernate.subclass
21   2317 aha     * discriminator-value="DPHImpl"
22   675  fers    * @hibernate.cache usage="read-write"
23   223  stankar */
24   223  stankar public class DPHImpl
25   1061 mila    extends DPH implements Historizable
26   223  stankar {
27   1019 fers        // concrete business methods that were declared
28   1019 fers    // abstract in class DPH ...
29   2317 aha
30   1019 fers        public static DPH getAktualniDPH() throws FWException
```

Kompilace

- Kompilace musí být:
  - snadno spustitelná
  - snadno konfigurovatelná
- Musí řešit:
  - závislosti na správných verzích knihoven
  - závislosti na prostředí (db, aplikační server, ...)
  - na všech systémech musí být postup stejný
  - konfigurace překladů (debug/release, různé varianty produktu, …)
- Kompilace přímo pomocí javac nebo IDE je nevhodná
  - není automatizovaná, není dávková

Kompilace

- Používané programy:
  - Visual Studio .NET
  - make
    - » autoconf/automake
    - » CMake
    - » ...
  - Ant
  - Maven
  - ...

Ant – build.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<project name="example" default="compile" basedir=".">
  <property name="app.name"      value="example"/>
  <property name="app.version"   value="1.0"/>
  <property name="build.home" value="${basedir}/target" />
  <target name="clean" description="Delete old build and dist directories">
    <delete dir="${build.home}" />
  </target>
  <target name="dist" depends="compile">
    <jar destfile="${build.home}/${app.name}-${app.version}.jar">
      <fileset dir="${build.home}/classes"><include name="**/*.*" />
      </fileset>
    </jar>
  </target>
  <target name="compile" description="Compile project">
    <javac srcdir="${basedir}/src/java" destdir="${build.home}/classes">
        <classpath><pathelement location="${basedir}/lib/log4j-1.2.8.jar" />
        </classpath>
    </javac>
  </target>
</project>
```

Ant

- Ant řeší jak co zkompilovat
  - velké množství targetů/maker
    - spuštění testů
    - spuštění noční kompilace
    - vygenerování jejich HTML reportů
    - přidání odkazu na reporty
    - umístění na web
    - ...

Ant vs. Maven

- Ant řeší jak co zkompilovat
  - musíte mít velké množství targetů/maker
    - puštění testů
    - puštění noční kompilace
    - vygenerování jejich HTML reportů
    - přidání odkazu na reporty
    - umístění na web
    - …
- Maven je nadstavba Antu
    - umožňuje používat vše z Anta
    - projekt je definován deklarativním způsobem
    - definuje standardní strukturu projektu
    - řeší závislosti na knihovnách, nástrojích

Maven genapp

```
fers@elara:~/ $ maven genapp

 __  __
|  \/  |__ _Apache__ ___
| |\/| / _` \ V / -_) ' \  ~ intelligent projects ~
|_|  |_\__,_|\_/\___|_||_|  v. 1.0.2

Enter a project template to use: [default]

Please specify an id for your application:  [app]
example
Please specify a name for your application:  [Example Application]
SoftEU example
Please specify the package for your application:  [example.app]
cz.softeu.example
build:start:

genapp:
    [copy] Copying 1 file to ./src/java/cz/softeu/example
    [copy] Copying 3 files to ./src/test/cz/softeu/example
    [copy] Copying 1 file to ./
    [copy] Copying 2 files to ./
BUILD SUCCESSFUL
Total time: 58 seconds
Finished at: Tue Oct 11 13:56:29 CEST 2005
```

Maven – project.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<project>
    <pomVersion>3</pomVersion>
    <name>SoftEU example</name>
    <groupId>softeu</groupId>
    <artifactId>example</artifactId>
    <version>1.0</version>
    <dependencies>
       <dependency>
            <groupId>log4j</groupId>
            <artifactId>log4j</artifactId>
            <version>1.2.8</version>
            <type>jar</type>
        </dependency>
    </dependencies>
    <build>
        <sourceDirectory>src/main/java</sourceDirectory>
    </build>
</project>
```

Maven

- `maven java:compile`
  - překompiluje projekt
- `maven dist`
  - vytvoří .jar a vygeneruje projektový web (viz další stránka)
- `maven eclipse`
  - vygeneruje z project.xml projekt pro Eclipse (není potřeba plugin)
  - existují pluginy pro Eclipse, NetBeans, IDEA, JBuilder, ...

Maven - web

Last published: 11 October 2005 | Doc for 1.0

built by maven.

Checkstyle Results

The following document contains the results of Checkstyle XML .

Summary

Files	Infos	Warnings	Errors
3	0	0	12

Files

Files	I	W	E
cz/softeu/example/App.java	0	0	11
cz/softeu/example/package.html	0	0	1

cz/softeu/example/App.java

Error	Line
Line does not match expected header line of '<?xml version="1.0" encoding="UTF-8"?>'.	1
Line has trailing spaces.	9
Utility classes should not have a public or default constructor.	9
'{' should be on the previous line.	10
Missing a Javadoc comment.	11

Bugtracking systémy

- Jednoduchá databáze chyb/nových vlastností
  - popis chyby/zadání práce k udělání
  - klasifikace – produkt, verze, priorita, vývojář

- Programátor po příchodu do práce řeší
  - své bugy
  - pokud něco objeví, zadá další bug
- Zvyšuje se koncentrace
  - dořeš to, co děláš, a zbytek odlož na potom
  - pokud předáš bug někomu jinému, nerušíš ho teď v jeho práci
  - nahrazuje papírky, které se mohou ztratit (když je někdo nemocný)

Bugtracking systémy

- Někdy rozlišujeme chyby a nové vlastnosti
- Při opravě chyby přidáme test, který testuje, zda jsme chybu opravili
- Chyby, které opravíme jsou pak kontrolovány nezávisle Q&A
    - verifikace opravy

- Používáme Bugzillu
    - do Bugzilly už dáváme i požadavky na koupi cukru a kávy
    - má výborný dotazovací systém
    - má závislosti mezi chybami
    - dnes je zřejmě nejpoužívanější

Testování

- Testy:
  - ruční
  - automatizované
  - testy použitelnosti
  - …

- Automatizované testy jsou pouštěny automatizovaně
  - při commitu
  - nočně
  - ...

Testování

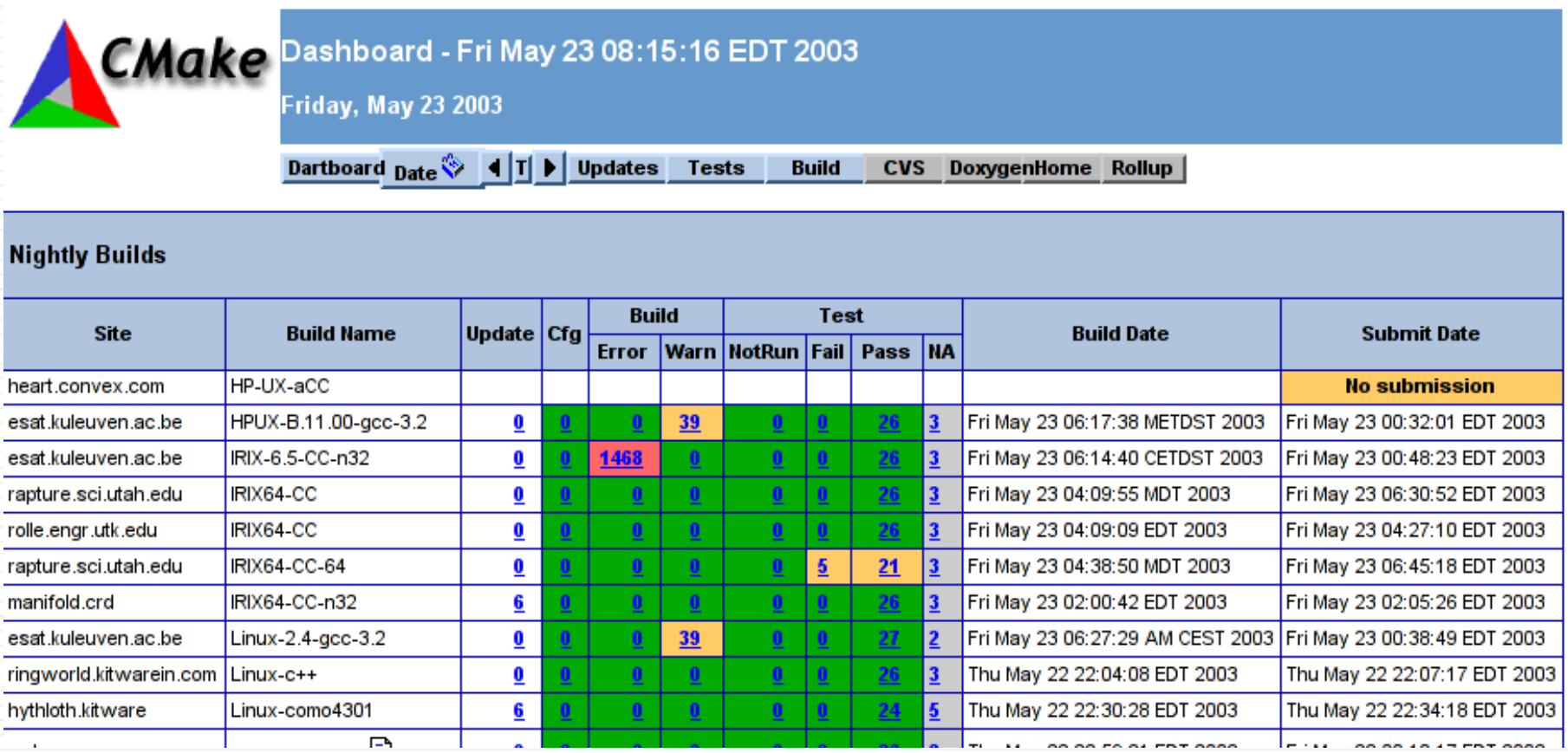
- Automatizované testy:
  - **unit testy**
  - **integrační testy**
  - **GUI testery** (nejsou moc obvyklé)
  - **statické analýzy kódu**
  - **výkonnostní testy** (jak se výkon měnil mezi verzemi)
  - **zátěžové testy**
  - …

Kontinuální integrace

- pravidelně spouštěná kompilace a testy. Výsledky jsou přístupné (včetně starších verzí)
- „hlídá“, že jde aktuální verze kompilovat a funguje
- velmi důležité při podpoře více platforem, překladačů, ...
    - program funguje na jedné platformě, ale nemusí na dalších
- obvykle vytváří i „instalačky“, které pak testují živí testeři
- používáno spíše u středních a větších projektů (více jak 5 lidí)
- bez ní není možné větší projekty dokončit

Kontinuální integrace

Ukázka výstupu programu dashboard



CMake

Dashboard - Fri May 23 08:15:16 EDT 2003

Friday, May 23 2003

Dartboard | Date | T | Updates | Tests | Build | CVS | DoxygenHome | Rollup

Nightly Builds

Site	Build Name	Update	Cfg	Build		Test				Build Date	Submit Date
				Error	Warn	NotRun	Fail	Pass	NA		
heart.convex.com	HP-UX-aCC										No submission
esat.kuleuven.ac.be	HPUX-B.11.00-gcc-3.2	0	0	0	39	0	0	26	3	Fri May 23 06:17:38 METDST 2003	Fri May 23 00:32:01 EDT 2003
esat.kuleuven.ac.be	IRIX-6.5-CC-n32	0	0	1468	0	0	0	26	3	Fri May 23 06:14:40 CETDST 2003	Fri May 23 00:48:23 EDT 2003
rapture.sci.utah.edu	IRIX64-CC	0	0	0	0	0	0	26	3	Fri May 23 04:09:55 MDT 2003	Fri May 23 06:30:52 EDT 2003
rolle.engr.utk.edu	IRIX64-CC	0	0	0	0	0	0	26	3	Fri May 23 04:09:09 EDT 2003	Fri May 23 04:27:10 EDT 2003
rapture.sci.utah.edu	IRIX64-CC-64	0	0	0	0	0	5	21	3	Fri May 23 04:38:50 MDT 2003	Fri May 23 06:45:18 EDT 2003
manifold.crd	IRIX64-CC-n32	6	0	0	0	0	0	26	3	Fri May 23 02:00:42 EDT 2003	Fri May 23 02:05:26 EDT 2003
esat.kuleuven.ac.be	Linux-2.4-gcc-3.2	0	0	0	39	0	0	27	2	Fri May 23 06:27:29 AM CEST 2003	Fri May 23 00:38:49 EDT 2003
ringworld.kitwarein.com	Linux-c++	0	0	0	0	0	0	26	3	Thu May 22 22:04:08 EDT 2003	Thu May 22 22:07:17 EDT 2003
hythloth.kitware	Linux-como4301	6	0	0	0	0	0	24	5	Thu May 22 22:30:28 EDT 2003	Thu May 22 22:34:18 EDT 2003

Kontinuální integrace

Pohled na detail s testy

Testing started on Thu May 22 23:29:20 EDT 2003

Site Name: KALGAN

Build Name: CYGWIN-c++

26 passed, 0 failed, 0 not run

Name (sort by)	Status	Time (sort by)	Detail
complex	Passed	28.366	
complexOneConfig	Passed	26.167	
conly	Passed	3.596	
curl	Passed	117.698	
dependency_w_libout	Passed	21.052	
dependency_wo_lib_out	Passed	22.279	

Debuggery a logování

- Debugger je přeceňován (ale je také důležitý)
- Mnoho věci rychleji odchytíme z logů
- Chybu hledejte v hlavě a ne v kódu
- Některé firmy zakazují používat debugger

- Logování umožní najít chybu zpětně (např. u zákazníka)
- Logování je možné konfigurovat za běhu
- Logování lze odeslat emailem

Profilery

- Neefektivní optimalizovat při psání kódu
  - přemýšlet při jeho psaní
- Optimalizovat pouze dlouhotrvající akce
- Profiler upozorní na problematické místo
- Profiler zanáší svým během chybu měření

- MiniProfilery – „levný“ profiler

Statické analýzy kódu

- U projektů je potřeba hlídat kvalitu kódu v rozumných mezích.
- Proto součástí kontinuální integrace je i generování různých statistik:
  - hledání duplicit kódu – Copy&Paste
  - hlídání použití některých konstrukcí (prázdný catch {})
  - hlídání dodržování coding style
  - hledání častých chyb, které projdou překladačem
  - pokrytí testy

Pokrytí testy

- angl. test coverage
- někdy je těžké poznat, které části programu testujete
- existují nástroje, které to dokáží hlídat
- při pouštění testů se generuje, který blok byl spuštěn -> reporty
- cílem je mít co největší pokrytí testy

- 100% pokrytí ještě neznamená, že nemáme chyby

  Příklad:

```
int pocitej(int a, int b)
{
return a / b;
}
```

- Při zavolání `pocitej(10, 2);` máme pokrytí 100%.
- Při zavolání `pocitej(1, 0);` dojde k chybě – dělení nulou.

Pokrytí testy

EMMA Coverage Report (generated Tue May 18 22:13:27 CDT 2004)

[all classes]

COVERAGE SUMMARY FOR PACKAGE [org.apache.velocity.runtime.parser]

name	class, %	method, %	block, %	line, %
org.apache.velocity.runtime.parser	88% (7/8)	82% (241/293)	58% (12411/21359)	58% (2421.5/4144

COVERAGE BREAKDOWN BY SOURCE FILE

name	class, %	method, %	block, %	line, %
TokenMgrError.java	0% (0/1)	0% (0/6)	0% (0/173)	0% (0/34)
VelocityCharStream.java	100% (1/1)	67% (16/24)	41% (393/952)	51% (85.8/169)
ParserTokenManager.java	100% (1/1)	80% (75/94)	57% (6034/10612)	55% (1159.7/2120)
Parser.java	100% (2/2)	91% (135/149)	61% (5547/9032)	64% (1088/1700)
ParseException.java	100% (1/1)	80% (4/5)	68% (258/377)	61% (39.9/66)
Token.java	100% (1/1)	67% (2/3)	75% (9/12)	75% (3/4)
JJTParserState.java	100% (1/1)	75% (9/12)	85% (170/201)	88% (45/51)

[all classes]

EMMA 2.0.4015 (stable) (C) Vladimir Roubtsov

Pokrytí testy

```
3194  final private int jj_ntk() {
3195    if ((jj_nt=token.next) == null)
3196      return (jj_ntk = (token.next=token_source.getNextToken()).kind);
3197    else
3198      return (jj_ntk = jj_nt.kind);
3199  }
3200
3201  private java.util.Vector jj_expentries = new java.util.Vector();
3202  private int[] jj_expentry;
3203  private int jj_kind = -1;
3204  private int[] jj_lasttokens = new int[100];
3205  private int jj_endpos;
3206
3207  private void jj_add_error_token(int kind, int pos) {
3208    if (pos >= 100) return;
3209    if (pos == jj_endpos + 1) {
3210      jj_lasttokens[jj_endpos++] = kind;
3211    } else if (jj_endpos != 0) {
3212      jj_expentry = new int[jj_endpos];
3213      for (int i = 0; i < jj_endpos; i++) {
3214        jj_expentry[i] = jj_lasttokens[i];
3215      }
3216      boolean exists = false;
3217      for (java.util.Enumeration enum = jj_expentries.elements(); enum.hasMoreElements();) {
3218        int[] oldentry = (int[])(enum.nextElement());
3219        if (oldentry.length == jj_expentry.length) {
3220          exists = true;
3221          for (int i = 0; i < jj_expentry.length; i++) {
3222            if (oldentry[i] != jj_expentry[i]) {
3223              exists = false;
3224              break;
```

Talkback

- Sbírání chyb u zákázníka
- Umožňuje časně reagovat na chyby
- Některé chyby nenasimulujete
- Ne všechny chyby zákazník sám reportuje
- Znáte z windows „Informovat vývojaře o chybě“

Sdílení znalostí

- každá firma má své „stříbro“ – znalosti.
- je třeba ukládat spoustu informací – návody, postupy, odkazy, …
- musí být snadno dostupné a musí být snadné je vytvářet
- vývojaři rádi uloží informace, když to bude jednoduché

- obvykle různé CMS – např. Wiki
  - my používáme TWiki

Nástroje se doplňují

- u chyby sledovat seznam změn, které jí opravují
- kontinuální integrace používá data z verzovacího systému
- jaká změna způsobila selhání kompilace a kdo ji provedl
- u kódu vidět, kdo jej napsal a kterou chybu opravuje – vlastnost přidává
- provázání systému automatické zpětné vazby s chybami v bug trackingu
- ...

Co potřebujete?

- **buildovací systém** (Ant, Make, Maven, ...)
- **verzovací systém** (CVS, SVN, BitKeeper, ...)
- **bug tracking systém** (Bugzilla, RT, ...)
- **kontinuální kompilace**
- **automatizované testování** (JUnit, CPPUnit)
- **systémy sdílení znalostí** (Wiki, nástěnka, Google, ...)
- **statické analyzátory kódu** (Checkstyle, ...)
- **profilovací nástroje**
- **generování a správa dokumentace**

- **velký hrnek na čaj, kafe; pizzu**
- **sluchátka**