



MASARYKOVA UNIVERZITA

## **PV213 Enterprise Information Systems in Practice**

### **05 – Security, Configuration management**



# MASARYKOVA UNIVERZITA

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.



INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.



INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

## Security overview

- ❏ Security is a big topic and will become even more important in the future
- ❏ We will concentrate just on often security holes in web applications
- ❏ Just remember that for security you have two approaches
  - ❏ Passive protection
    - ❏ “Classical” protection (firewalls, accounts, rights, ...)
  - ❏ Active protection
    - ❏ Proactive finding security holes (tools, security audits, ...)
- ❏ Securing systems requires knowledge
  - ❏ Architects, developers and testers should continuously learn
- ❏ Applied security protection should correspondent with importance of the system (e.g. bank application vs. public blog)
- ❏ Keep in mind that there isn't 100% secure system

## The Open Web Application Security Project (OWASP)

- ❏ OWASP is non-profit organization focused on improving the security of application software
- ❏ Provides documentation and tools regarding security
  - ❏ Protection
  - ❏ Detection
  - ❏ Life cycle
- ❏ Started to work in 2003
- ❏ Provides Top 10 security risks in years 2004, 2007, 2010, 2013
- ❏ Following slides show OWASP Top 10 for 2013 (release candidate)
- ❏ <https://www.owasp.org>



## A1: Injection

- ❏ Injection is inclusion of malicious data into the input of the application which causes then some weakness in the application
- ❏ There are several types of injections: SQL, OS Shell, ...
- ❏ SQL injection example
  - ❏ `String query = "SELECT * FROM customers WHERE customerId = " + request.getParameter("id")`
- ❏ What will happen when as "id" is passed following?
  - ❏ `0 OR 1 = 1`
- ❏ How to avoid
  - ❏ Use appropriate API (e.g. prepared statements instead of string concatenation for SQL)
  - ❏ Sanitize input (escape all special characters)
  - ❏ Use some white list approach

## A2: Broken Authentication and Session Management

- ❏ Occurs in cases e.g. when session id is part of the URL
- ❏ Example
  - ❏ <https://example.com/buy.html?sessionid=32263D8EFw49w6Jk>
- ❏ What will happen when such a link is send e.g via mail?
  - ❏ If there isn't another security check (e.g. check of IP address) any receiver can take over the session (and e.g. pay with your card)
- ❏ How to avoid
  - ❏ Don't store session identification in URL
  - ❏ Use session timeouts
  - ❏ Guarantee that session id is always unique

## A3: Cross-Site Scripting (XSS)

- ❏ XSS is insertion of malicious client-side script into the web page
- ❏ Such a script can e.g. steal secure data or user's current session
- ❏ Example
  - ❏ `<input name="comment" type="text" value="[read from DB]">`
- ❏ What will happen when “[read from DB]” is following?
  - ❏ `Hi!"<script>alert(document.cookie);</script><input type="hidden" value="`
- ❏ On the output will appear
  - ❏ `<input name="comment" type="text" value="Hi!"><script>alert(document.cookie);</script><input type="hidden" value="">`
- ❏ How to avoid
  - ❏ Sanitize output to avoid execution of the script
  - ❏ Use some white list approach



## A4: Insecure Direct Object References

- ❏ Insecure Direct Object References means that attacker can access data of someone else
- ❏ This problem occurs when there are missing security checks on the server side (after authentication application believes input parameters)
- ❏ Example
  - ❏ <https://example.com/viewAccount?custId=1234>
- ❏ What will happen when attacker simply changes custId?
  - ❏ He can see account of another customer
- ❏ How to avoid
  - ❏ Eliminate direct reference (e.g. use temporary random mapping)
  - ❏ Validate direct reference (verify user is allowed to do this task)

## A5: Security Misconfiguration

- ❏ There can be several reasons for this attack e.g.
  - ❏ You use some default configuration which is not secure
    - ❏ Installed system services you don't use
    - ❏ Installed administrative applications you don't need
    - ❏ Used default passwords
  - ❏ You report more than required to the end user (stack traces in case of failure, directory listing, etc.)
- ❏ How to avoid
  - ❏ Use latest patches for all components (OS, framework, ...)
  - ❏ Don't rely on default configuration
  - ❏ Use as few components as possible

## A6: Sensitive Data Exposure

- ❑ Sensitive data are stored on the unsecured (or inefficiently secured) storage or transported via unsecured channel
  - ❑ Credit card numbers stored in the database in plain text
  - ❑ Passwords stored encrypted but without using salt
  - ❑ Application doesn't use SSL or use it only for some pages
  - ❑ Application uses weak cryptographic algorithms or keys
- ❑ How to avoid
  - ❑ Don't store sensitive data at all
  - ❑ Store sensitive data encrypted and only for minimal time
  - ❑ Use standard strong encryption algorithms
  - ❑ Protect passwords / certificates, keep certificates up-to-date
  - ❑ Use secure transport channels

## A7: Missing Function Level Access Control

- ❏ URLs to resources are not protected on the server side
  - ❏ Similar problem to Insecure Direct Object References
- ❏ Occurs in cases when access to given resource is not authorized on the server side (authorization is done only via hiding data)
- ❏ Example
  - ❏ <https://example.com/user/viewAccount>
  - ❏ <https://example.com/admin/viewAccount>
- ❏ How to avoid
  - ❏ Do the resource authorization on the server side
  - ❏ Restrict access only to required resources (use white list approach) - e.g. disallow accessing logs, listing directories, etc.
  - ❏ Check your server configuration - disallow all by default

## A8: Cross-Site Request Forgery (CSRF)

- ❏ CSRF is all about default behavior of browsers which automatically provide most credentials with each request (session cookie, basic authentication header, etc.)
- ❏ Example
  - ❏ User logs into the banking application (victim site)
  - ❏ Attacker at this time instructs user to view attackers site (via mail, etc.)
  - ❏ Attacker's site contains (in hidden image) link to the victim site e.g. for transferring money
  - ❏ Such a link represents GET request to the victim site and browser sends also credentials (cookies etc.)
  - ❏ Victim site cannot distinguish from where request was sent
- ❏ How to avoid
  - ❏ Use POST requests instead of GET requests for all important actions
  - ❏ Put random authorization token into each form which is send by POST request (attacker cannot guess this token)

## A9: Using Known Vulnerable Components

- ❑ Nearly all software build today uses for some of its sub-functionality third-party components
- ❑ These components can contain security issues
- ❑ Attackers concentrate on often used components
- ❑ Sometimes security patches are not available for older versions of components (you have to install new versions to apply the patch)
- ❑ How to avoid
  - ❑ Know all your components including their dependencies
  - ❑ Periodically monitor security news for these components
  - ❑ Define policies e.g. that components have to pass security tests
  - ❑ Apply security patches and keep your components up-to-date

## A10: Unvalidated Redirects and Forwards

- ❏ Application allows to redirect or forward to another page via supplied parameter but there isn't check what is allowed
- ❏ Example
  - ❏ <https://example.com/redirect?url=https://evil.com>
- ❏ What will happen when user will be redirected?
  - ❏ Attacker can simulate the functionality of the application and steal sensitive information
- ❏ How to avoid
  - ❏ Avoid using redirects and forwards at all (not always possible)
  - ❏ Avoid using parameterized redirects and forwards
  - ❏ Check input parameters (e.g. use mapping for parameters instead of directly specifying target URL)

## OWASP Top 10 for 2010

A1: Injection

A2: Cross-Site Scripting (XSS)

A3: Broken Authentication and Session Management

A4: Insecure Direct Object Reference

A5: Cross Site Request Forgery (CSRF)

A6: Security Misconfiguration

A7: Insecure Cryptographic Storage

A8: Failure to Restrict URL Access

A9: Insufficient Transport Layer Protection

A10: Unvalidated Redirects and Forwards



## OWASP Top 10 for 2007

A1: Cross Site Scripting (XSS)

A2: Injection Flaws

A3: Malicious File Execution

A4: Insecure Direct Object Reference

A5: Cross Site Request Forgery (CSRF)

A6: Information Leakage and Improper Error Handling

A7: Broken Authentication and Session Management

A8: Insecure Cryptographic Storage

A9: Insecure Communications

A10: Failure to Restrict URL Access

## OWASP Top 10 for 2004

A1: Unvalidated Input

A2: Broken Access Control

A3: Broken Authentication and Session Management

A4: Cross Site Scripting

A5: Buffer Overflow

A6: Injection Flaws

A7: Improper Error Handling

A8: Insecure Storage

A9: Application Denial of Service

A10: Insecure Configuration Management

## OWASP Top 10 Mobile Risks, Release Candidate v1.0 (Sep. 2011)

- M1: Insecure Data Storage
- M2: Weak Server Side Controls
- M3: Insufficient Transport Layer Protection
- M4: Client Side Injection
- M5: Poor Authorization and Authentication
- M6: Improper Session Handling
- M7: Security Decisions Via Untrusted Inputs
- M8: Side Channel Data Leakage
- M9: Broken Cryptography
- M10: Sensitive Information Disclosure

## Configuration management overview I

### What is Configuration management

- ❏ Tracking and controlling changes in the software
- ❏ Way how you control evolution of the SW project
- ❏ Tools and processes for controlling the SW project

### What everything it contains

- ❏ Management of tools for
  - ❏ Requirements management
  - ❏ Project planning
  - ❏ Version control (for source code and documents)
  - ❏ Development
  - ❏ Building
  - ❏ Testing
  - ❏ Bug tracking
  - ❏ ...

## Configuration management overview II

### What everything it contains (continue)

- ❏ Processes for
  - ❏ Building, continuous integration
  - ❏ Branching of source code, merging of source code
  - ❏ Management of versioning
  - ❏ Releasing the SW (deployment), migration
  - ❏ Backup of source code, documentation, released versions etc.
  - ❏ ...

Configuration manager is role in the team and he is responsible for creation and maintaining **configuration management plan** where should be described all supporting tools and processes.

## Version control systems - Motivation

**Q: How do you store the source code?**

❏ A: Just locally

❏ Q: How do you cooperate with other people?

❏ A: On the shared drive

❏ Q: How do you keep history of changes?

❏ Q: How do you know who made appropriate change?

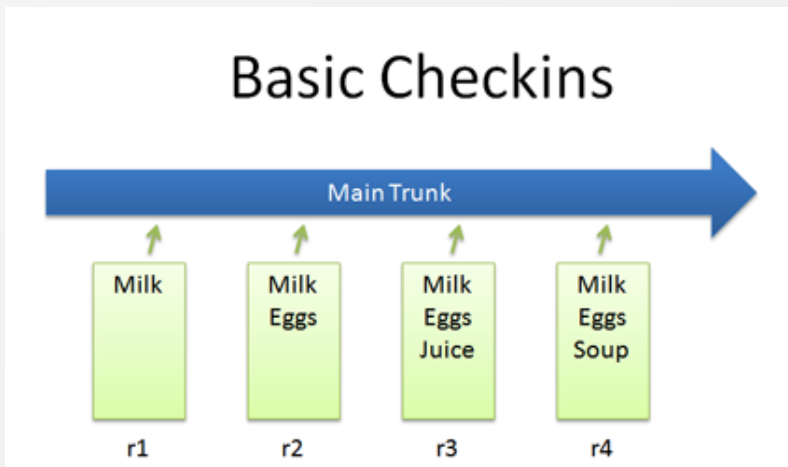
❏ Q: How do you do merges of changes from different developers?

❏ Q: How do you manage to do maintenance of the already released version and development for new version in parallel?

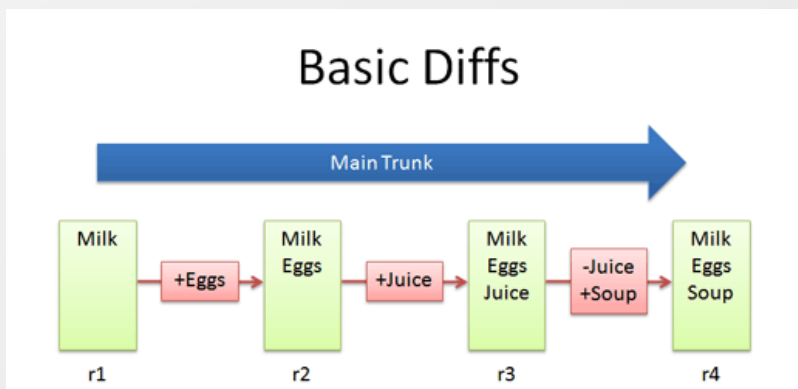
Version control systems are not only for source code (e.g. also for documentation) but mainly when people speak about version control system they mean versioning of the source code.

# Version control systems - Checkout / Checkin

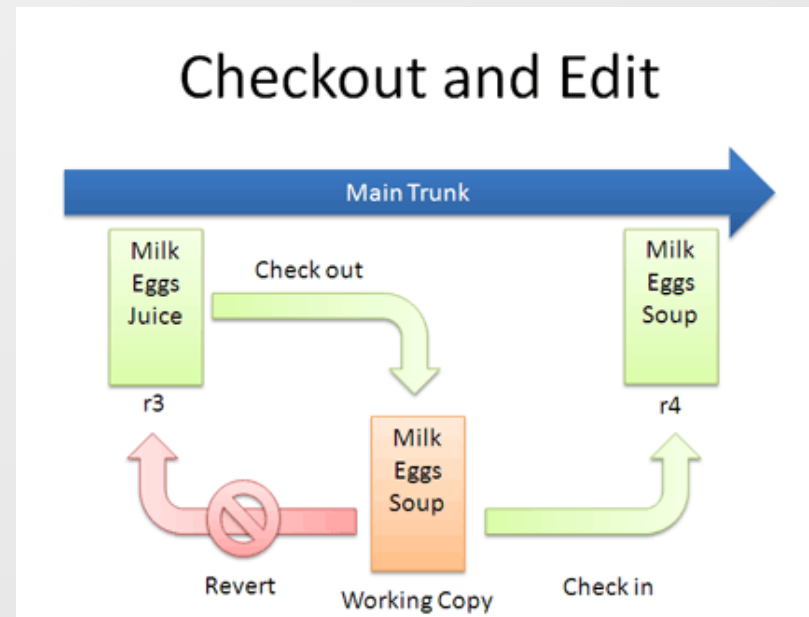
## Basic Checkins



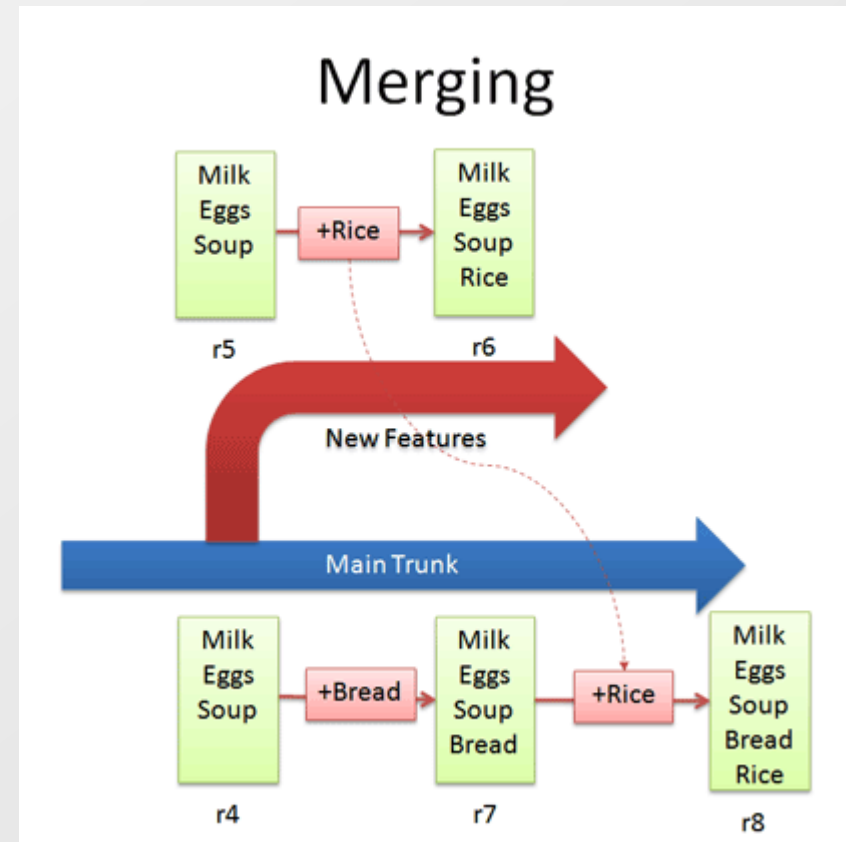
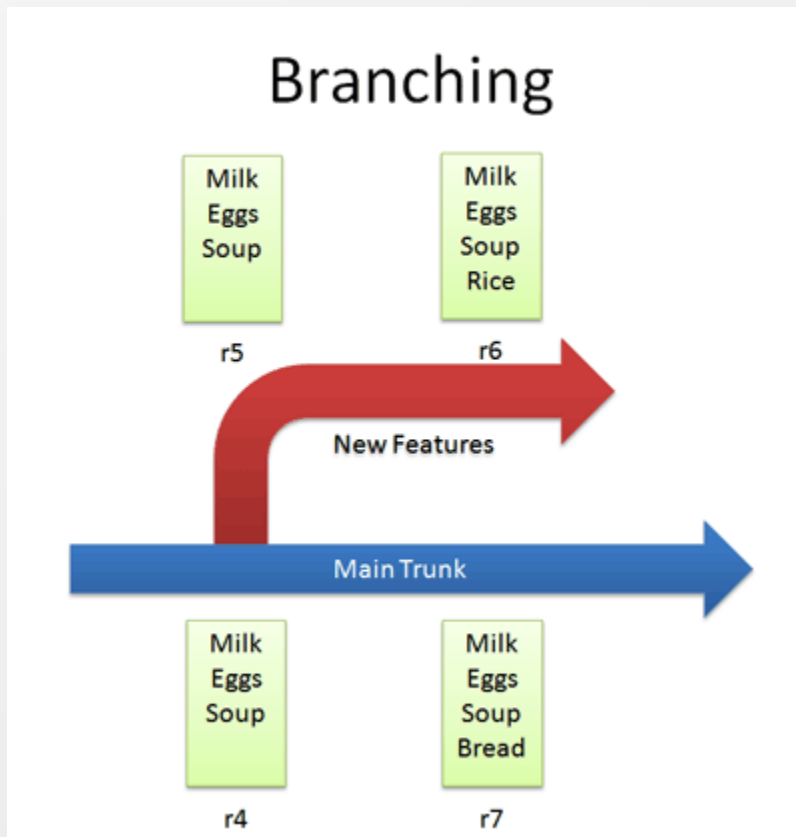
## Basic Diffs



## Checkout and Edit

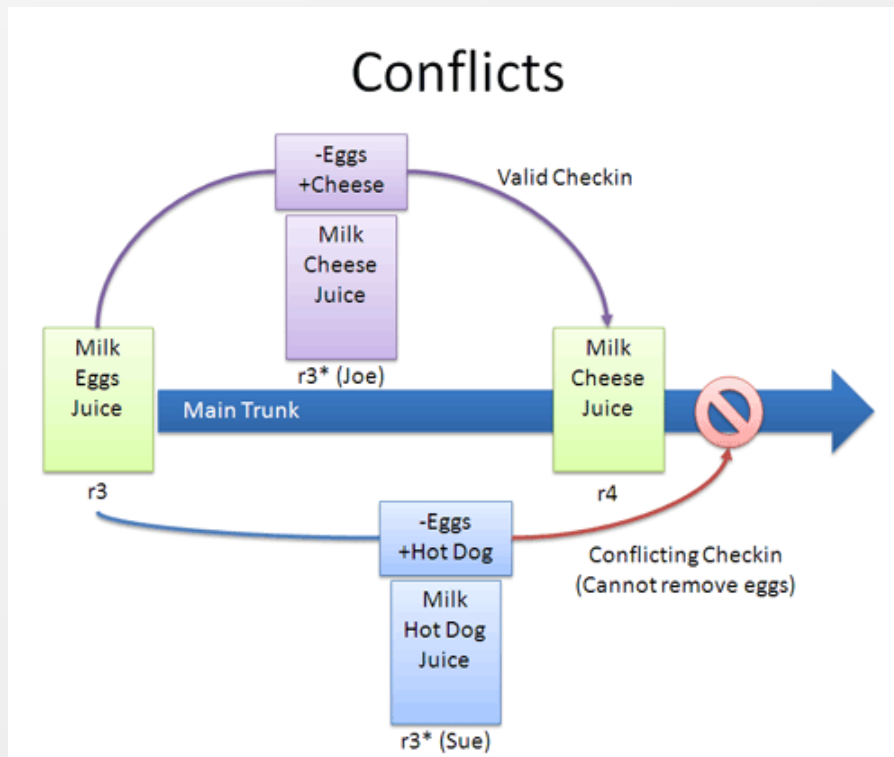


## Version control systems - Branching and merging



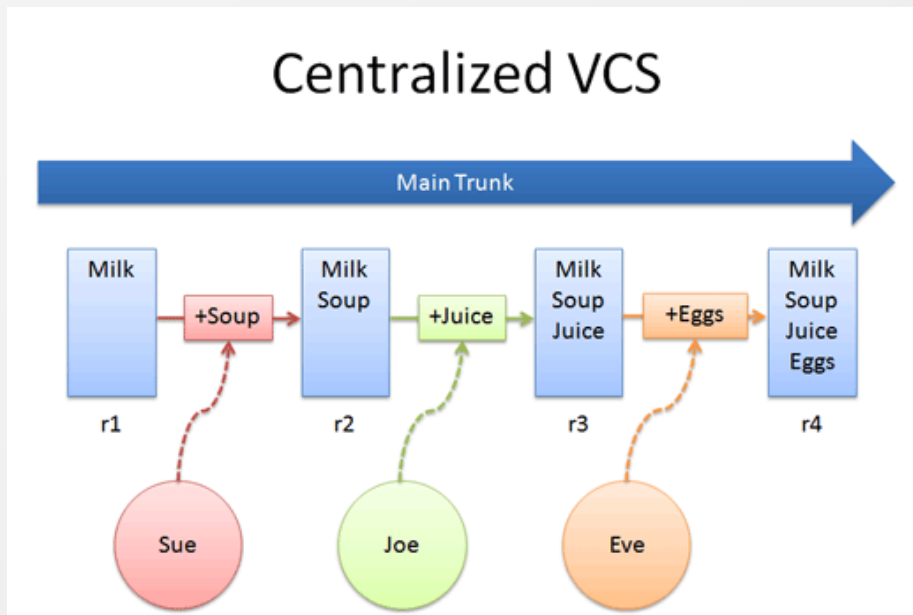


# Version control systems - Solving conflicts, tagging

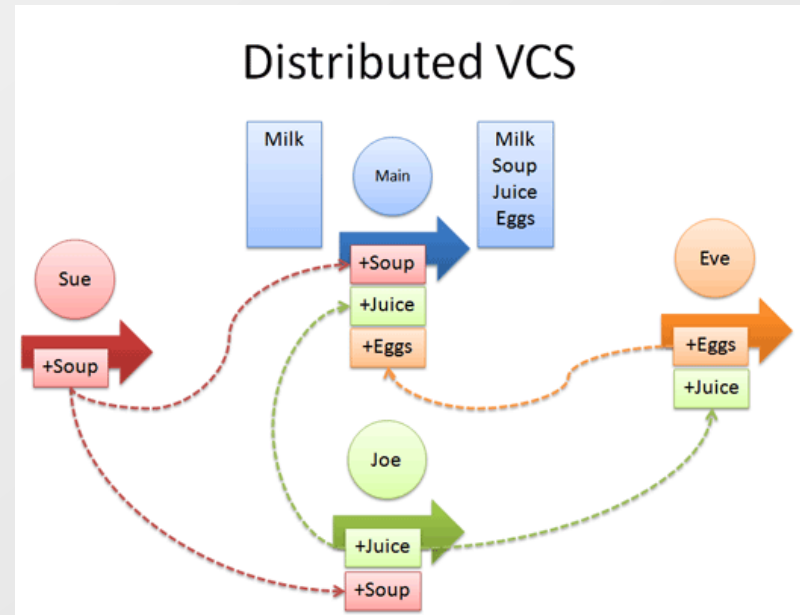


# Version control systems - Centralized vs. distributed

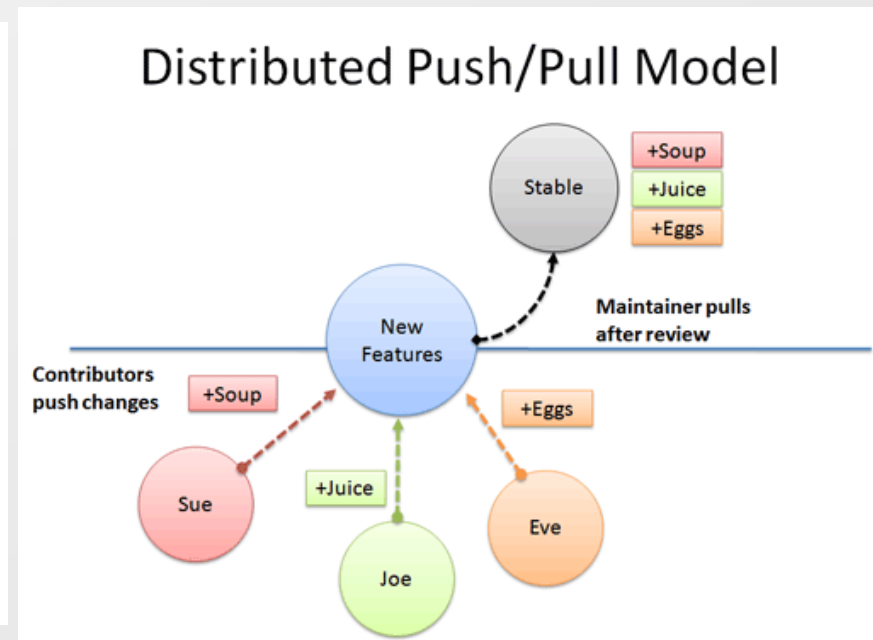
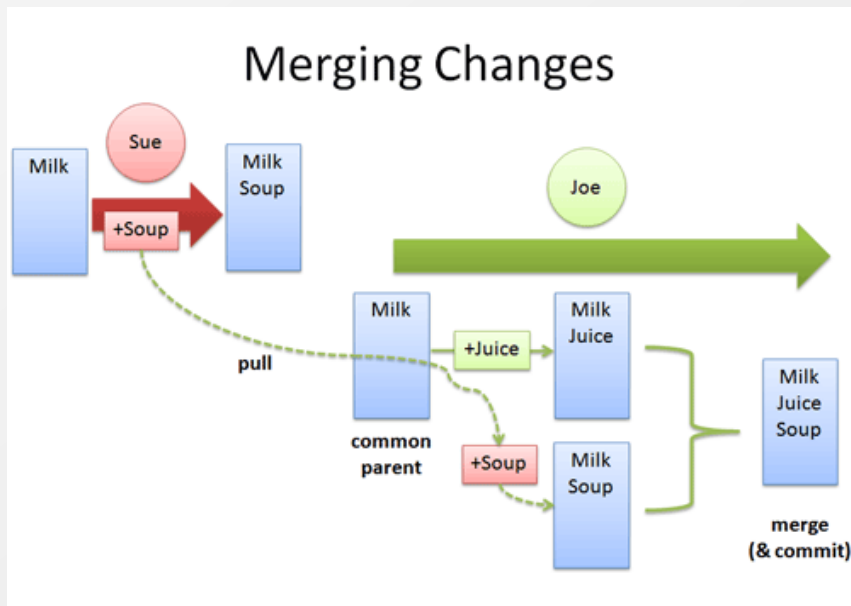
Centralized VCS



Distributed VCS



# Version control systems - Distributed approach



## Version control systems - Available tools I

### CVS (Concurrent Version System)

- ❑ Old system which is still used (first version in late 1980s)
- ❑ Has some bad functionality by design
  - ❑ No support for atomic commits
  - ❑ No support for versioning of directories
  - ❑ Rather complicate branching

### Subversion (SVN)

- ❑ Different approach for versioning - applies to the entire tries (not to individual files)
- ❑ Supports atomic transactions
- ❑ No support for merge of renamed files

## Version control systems - Available tools II

### Git, Mercurial, Bazaar

- ❑ Distributed version control systems
- ❑ Every user has it's own copy of the repository
- ❑ You can commit your changes offline (repositories are synchronized when you are online again)

### ClearCase

- ❑ Commercial from IBM (formerly Rational)
- ❑ Distributed version control system (several replicas)
- ❑ Every file or directory is owned by specific replica
- ❑ Dynamic and snapshot views

### Team Foundation Server

- ❑ Commercial from Microsoft
- ❑ Successor of Visual SourceSafe

## Version control systems - ClearCase Configuration Specification

```
# Show all elements that are checked out to this view, regardless any other rules.
element * CHECKEDOUT

# If an element has a version on the 'module2_dev_branch', then the latest
# version of this branch shall be the visible version in this view.
element * ../module2_dev_branch/LATEST

# For all files named 'somefile', regardless of location, always show the latest version
# on the main branch.
element ../somefile /main/LATEST

# Use a specific version of a specific file. Note: This rule must appear before
# the next rule to have any effect!
element /vobs/project1/module1/a_header.h /main/proj_dev_branch/my_dev_branch1/14

# For other files in the 'project1/module1' directory, show versions
# labeled 'PROJ1_MOD2_LABEL_1'. Furthermore, don't allow any checkouts in this path.
element /vobs/project1/module1/... PROJ1_MOD2_LABEL_1 -nocheckout

# Show the 'ANOTHER_LABEL' version of all elements under the 'project1/module2' path.
# If an element is checked out, then branch that element from the currently
# visible version, and add it to the 'module2_dev_branch' branch.
element /vobs/project1/module2/... ANOTHER_LABEL -mkbranch module2_dev_branch
```

## Version control systems - Branching approach

### New development in main branch (head, trunk)

- ❑ All new features are committed to the main branch
- ❑ Next version of the application is developed in the main branch

### Released version in branch

- ❑ Branch is created during release process for the version
- ❑ In the branch is done only bug fixing for the current version

### Experimental (new) features in feature branches

- ❑ New features (especially for which it is not clear whether there will be contained in the next version or it is too risky to directly include them e.g. because of unclear time plan) special feature branches are used

In reality sometimes there is new functionality also in bug fix branches for already released version (especially for VCSes with poor branch support)

## Version control systems - Advanced features

For most version control systems you can define actions which are executed when given trigger is fired (mostly pre-commit / post-commit actions)

These actions can be used e.g. for:

- ❏ Forcing developers to provide comment or some ticket number which correspond with requirement
- ❏ Doing some changes in committed source (e.g. applying auto-formatting)
- ❏ Checking that there aren't known issues in the code
  - ❏ Checking naming conventions, absence of comments for public methods
  - ❏ Checking basic architectural rules
- ❏ Checking that with changed sources all tests still pass
- ❏ Checking that there exists unit tests for committed sources and there is at least 80% of code coverage



## Integrated Development Environment - IDE

- **IDE simplifies development tasks and increases productivity and quality**
- The same IDE used by all developers decreases maintenance costs
- **Examples of IDEs**
  - **Microsoft Visual Studio**
    - Commercial but basic variant is available for free
    - Targeted for Microsoft platform (desktop, mobile, cloud)
  - **Eclipse**
    - Open source (originally developed in IBM)
    - Goal is to be a platform for building applications (not only IDEs)
    - Originally targeted mainly for Java but plugins for lot of different languages exists these days
  - **NetBeans**
    - Open source supported by Oracle (formerly Sun)
    - Mainly IDE for Java

## Build process

Goal of the build process is to have build easily reproducible on any computer with minimal dependencies (without development environment).

Tools uses some type of the specialized scripting language which tells what and how sources are build and how build results are created.

### Possible tools:

- ❏ Java
  - ❏ Ant
  - ❏ Maven
- ❏ .NET
  - ❏ MSBuild
- ❏ C/C++
  - ❏ make

```
<!-- Ant example -->
<project name="Client" basedir="." default="compile-core">
  <target name="compile-core" depends="prepare">
    <echo>Compiling project sources...</echo>
    <javac srcdir="${src.dir}"
          destdir="${build.classes.dir}"
          target="1.6"
          source="1.6"
          classpathref="classpath"
          includeAntRuntime="false"
          debug="true">
      <include name="**" />
      <exclude name="**/gui/**" />
    </javac>
  </target>
</project>
```

## Continuous integration I

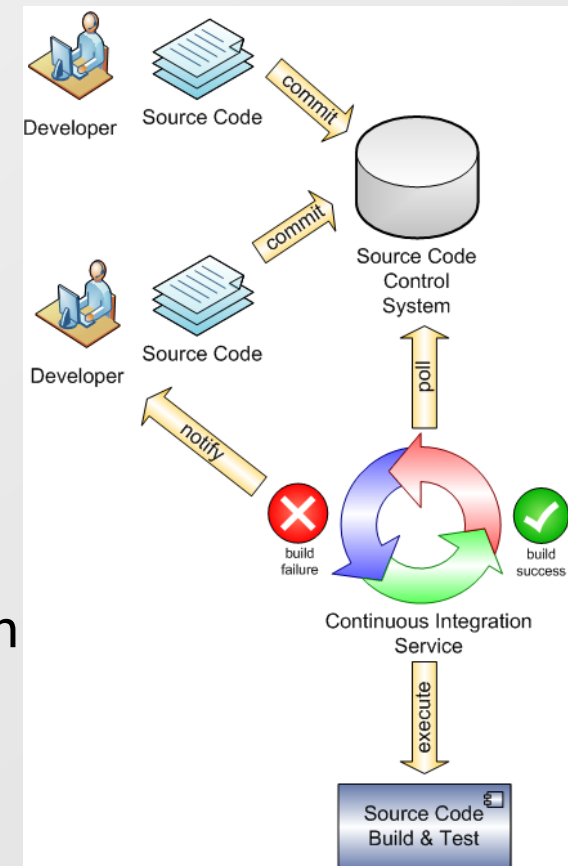
**Continuous integration** is approach how to integrate source code from different developers as soon as possible to early identify possible problems.

- ❏ Continuous integration is standard technique in agile development methods but you can apply it to any project (even when there is only one developer working on the project)
- ❏ Continuous integration helps with minimizing time needed for integration of different component from different developers
  - ❏ Detects whether the whole project is still compilable (e.g. some forgotten commits)
  - ❏ Detects if all tests still pass
  - ❏ Reduces time needed for building and eliminates mistakes made by people during building

## Continuous integration II

### How it works

- ❏ Developers put their code into VCS
- ❏ Continuous integration server periodically checks if there are any changes in the CVS
- ❏ When change is detected it starts the build
- ❏ When build fails appropriate persons (developers / managers) are informed
- ❏ When build is successful results are stored
- ❏ Build results can be automatically deployed in the test environment
- ❏ After the successful build can be executed automatic tests



## Tools for bug tracking, project management, etc.

### Bugzilla

- ❑ Bug tracking system, open source

### Atlassian JIRA

- ❑ Bug and issue tracking and project management system
- ❑ Commercial but free for open source projects

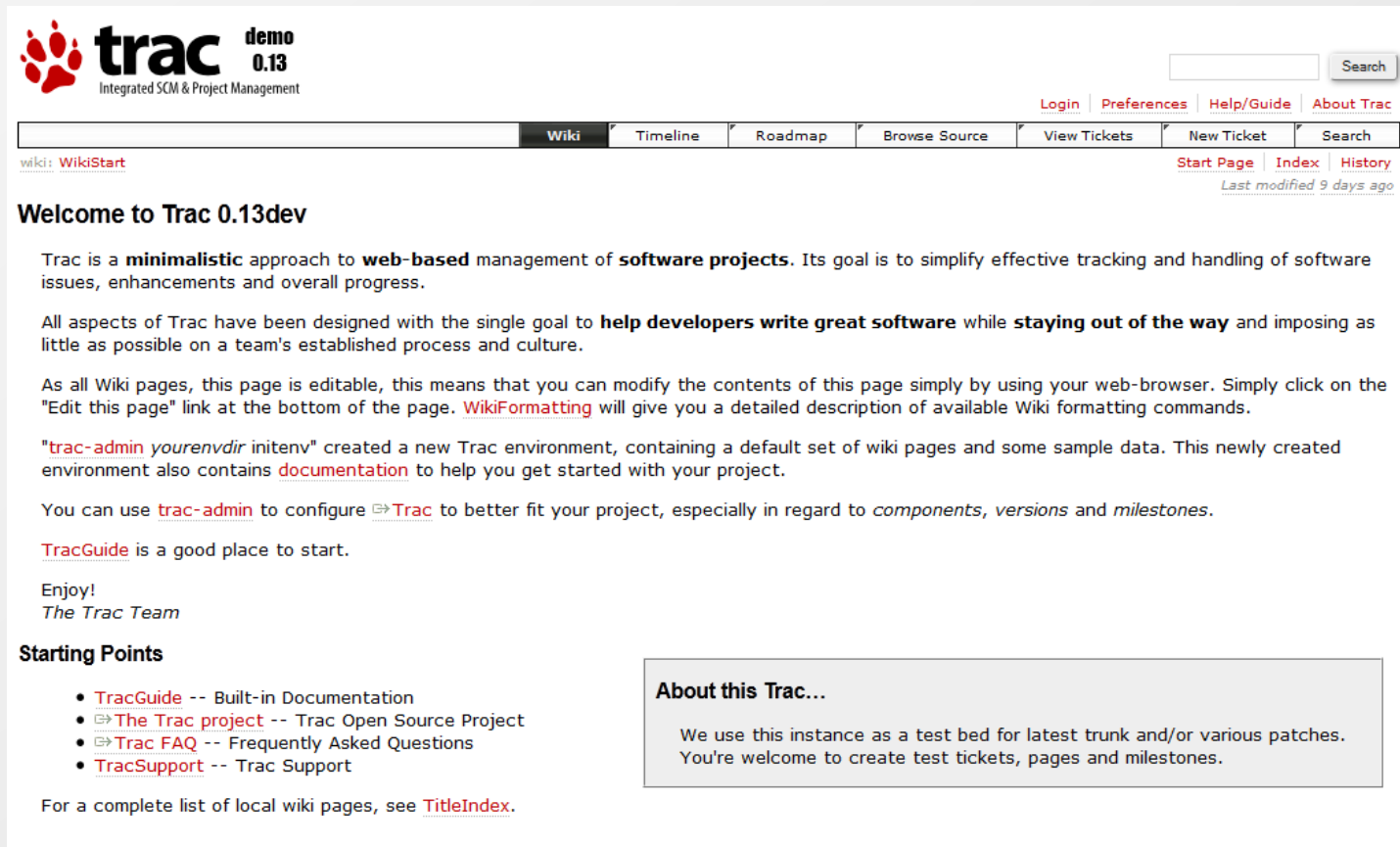
### Microsoft Team Foundation Server

- ❑ Bug and task tracking, quality control, project management and reporting system (among other features)
- ❑ Commercial but limited (maximal 5 users) free version announced in Team Foundation Server 2012 (TFS 11)

### TRAC

- ❑ Bug and task tracking, wiki, project management system
- ❑ Open source, a lot of additional plugins exists

# TRAC tool - Wiki



**trac demo 0.13**  
Integrated SCM & Project Management

[Login](#) | [Preferences](#) | [Help/Guide](#) | [About Trac](#)

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

wiki: [WikiStart](#) [Start Page](#) | [Index](#) | [History](#)  
Last modified 9 days ago

## Welcome to Trac 0.13dev

Trac is a **minimalistic** approach to **web-based** management of **software projects**. Its goal is to simplify effective tracking and handling of software issues, enhancements and overall progress.

All aspects of Trac have been designed with the single goal to **help developers write great software** while **staying out of the way** and imposing as little as possible on a team's established process and culture.

As all Wiki pages, this page is editable, this means that you can modify the contents of this page simply by using your web-browser. Simply click on the "Edit this page" link at the bottom of the page. [WikiFormatting](#) will give you a detailed description of available Wiki formatting commands.

"[trac-admin yourenvdir](#) initenv" created a new Trac environment, containing a default set of wiki pages and some sample data. This newly created environment also contains [documentation](#) to help you get started with your project.

You can use [trac-admin](#) to configure [Trac](#) to better fit your project, especially in regard to *components, versions* and *milestones*.

[TracGuide](#) is a good place to start.

Enjoy!  
*The Trac Team*

### Starting Points

- [TracGuide](#) -- Built-in Documentation
- [The Trac project](#) -- Trac Open Source Project
- [Trac FAQ](#) -- Frequently Asked Questions
- [TracSupport](#) -- Trac Support

For a complete list of local wiki pages, see [TitleIndex](#).

#### About this Trac...

We use this instance as a test bed for latest trunk and/or various patches. You're welcome to create test tickets, pages and milestones.

# TRAC tool - Timeline

**trac demo 0.13**  
Integrated SCM & Project Management

Search [ ]

[Login](#) | [Preferences](#) | [Help/Guide](#) | [About Trac](#)

Wiki | **Timeline** | Roadmap | Browse Source | View Tickets | New Ticket | Search

← Previous Period | Next Period →

### Timeline

**2012-04-14: Today**

- 10:45 Ticket #605 (Testksksksksks) created by anonymous  
ssssasasasasas
- 00:32 Ticket #604 (I've got a huge problem down under) created by anonymous  
*'The table*
- 00:06 [TracWiki](#) edited by anonymous  
(diff)
- 00:04 [TracWiki](#) edited by anonymous  
(diff)
- 00:02 [TracWiki](#) edited by anonymous  
(diff)

**2012-04-13: Yesterday**

- 23:40 Ticket #603 (Nem működik a menü) created by anonymous  
üdv nem működik az oldalon a menü.
- 23:07 Changeset [11032] by psuter  
1 edit in trunk/trac/perm.py  
0.13dev: Cache the permissions table. Related to #4245.
- 23:07 Changeset in trac.hg [7909:c03c677c3c26] trunk tip by psuter  
1 edit in trac/perm.py  
0.13dev: Cache the permissions table. Related to #4245.

View changes from 2012-04-14  
and 7 days back  
done by [ ]

Changesets in all repositories  
 Milestones reached  
 Tickets opened and closed  
 Ticket updates  
 Wiki changes

Update

# TRAC tool - Roadmap

**trac demo 0.13**  
Integrated SCM & Project Management

Search

[Login](#) | [Preferences](#) | [Help/Guide](#) | [About Trac](#)

Wiki | Timeline | **Roadmap** | Browse Source | View Tickets | New Ticket | Search

## Roadmap

Show completed milestones  
 Hide milestones with no due date

**Milestone: milestone1**  
No date set

23%

Total number of tickets: 209 - closed: 49 - active: 160

**Milestone: milestone2**  
No date set

25%

Total number of tickets: 69 - closed: 17 - active: 52

**Milestone: milestone3**  
No date set

22%

Total number of tickets: 36 - closed: 8 - active: 28



# TRAC tool - View tickets

**trac demo 0.13**  
Integrated SCM & Project Management

Search [input] [Search]

[Login](#) | [Preferences](#) | [Help/Guide](#) | [About Trac](#)

Wiki | Timeline | Roadmap | Browse Source | **View Tickets** | New Ticket | Search

[Available Reports](#) | [Custom Query](#)

**{1} Active Tickets** (481 matches)

- List all active tickets by priority.
- Color each row based on priority.

Max items per page:  [Update]

**Results (1 - 100 of 481)**

1 2 3 4 5 →

Ticket	Summary	Component	Version	Milestone	Type	Owner	Status	Created
#507	Tasks	component1			defect	rafal	new	2012-01-30
#528	hola mundo	component1	1.0		defect	tu mama	new	2012-02-13
#564	Standard-Defekt	component1			defect		new	2012-03-20
#595	I am not receiving emails.	component1			defect		new	2012-04-09
#532	Bitte erledigen ... aber ASAP	component1	2.0		task		new	2012-02-24
#545	Unable to create account	component1	1.0	milestone1	defect	Leye	new	2012-03-02
#554	this is testy test	component1	1.0	milestone1	defect	Hugo	new	2012-03-12
#555	Wohoo	component1	2.0	milestone1	defect	fdg	new	2012-03-12
#586	Bitte Projektplan liefern	component1		milestone2	task	karsten	new	2012-04-03
#260	Test	component2	2.0	milestone3	defect		new	2011-06-30
#362	Testing the new ticket creation	component2	2.0	milestone3	enhancement		new	2011-09-14
#409	MuratDeneme	component1	1.0		defect	Murat Kisakurek	new	2011-10-19
#567	ticket issue	component2			defect		new	2012-03-21
#580	ITMX msg resp error	component1	1.0		defect	mod	new	2012-04-01
#116	check my code	component2	2.0		task	anonymous	assigned	2011-03-16

# TRAC tool - Create new ticket

**trac** demo 0.13  
Integrated SCM & Project Management

Search

[Login](#) | [Preferences](#) | [Help/Guide](#) | [About Trac](#)

Wiki | Timeline | Roadmap | Browse Source | View Tickets | **New Ticket** | Search

### Create New Ticket

**Properties**

Summary:

Description:  You may use [WikiFormatting](#) here.

Type:  (dropdown menu open showing: defect, enhancement, task)

Milestone:  (dropdown menu)

Version:  (dropdown menu)

Priority:  (dropdown menu)

Component:  (dropdown menu)

Keywords:

Cc:

Release Notes:

API Changes:

Owner:

**Reporter**

Your email or username:

*E-mail address and user name can be saved in the [Preferences](#).*

I have files to attach to this ticket

# Děkuji za pozornost.

Tento projekt je spolufinancován Evropským sociálním fondem a státním rozpočtem České republiky.



INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ