



**OpenOffice.org**

Conference 2007 Barcelona

Jens-Heiner Rehtien, Sun Microsystems



OpenOffice.org  
Release  
Engineering

# Child workspaces and SCM

What do we require from our next  
SCM (Software Configuration  
Management) tool?

- CVS is aging ...
- The requirements for a new SCM tool
- Three strong contenders ...
  - Mercurial
  - Git
  - SubVersion
- ... and why we didn't choose one, yet
- Meanwhile: easing the pain with PCVSLib

## CVS is aging, and it shows ...

- tagging and branching are  $O(n)$  operations
- history preserving renames are awkward
- binary file handling is error prone
- commit operations are not atomic
- need to be online for all operations

## The size of the problem (as of Sep. 2007)

- repository size: 17 GiB
- files: 213000
- directories: 27500
- top level projects: 141
- code modules: > 200
  - > 8 GiB in size , > 1.2 million revisions
- tags: ~ 5000, branches: ~ 5000 (~ 500 live)

# The requirements for a new SCM tools

OOo uses the CWS development process

=> **SCM tool must fit to the CWS model**

Full requirement list:

[http://wiki.services.openoffice.org/wiki/SCM\\_Requirements](http://wiki.services.openoffice.org/wiki/SCM_Requirements)

# Mandatory requirements

- stable, proved
- clients for all our platforms
- must support all standard features of CVS
  - status, diff, log, annotate etc, as fast as CVS
  - proper support for binaries
- easily support branches and tags
  - lightweight (fast), support for rebasing
- preserve 7 years of OOo history on import

- a centralized SCM must provide
  - proper authentication framework
  - replication
- distributed SCM must provide
  - publishing framework

## Other important features ...

- drop in replacement for CVS
- easy integration into develop. framework
  - commit messages, connect commits with issues ...
- easy scriptable interface or client libraries
- easily traverses firewalls
- web browsable
- tinderbox and bonsai integration ...

# Three strong contenders ...

- **Mercurial** (<http://www.selenic.com/mercurial/wiki>)
  - DSCM, programmed in python and C
  - adequate documentation
- **Git** (<http://git.or.cz>)
  - DSCM, fast, good rebase support
  - tool based approach
- **SubVersion** (<http://subversion.tigris.org>)
  - CSCM, nearest replacement for CVS, fast
  - good documentation
  - nice client libraries (bindings for many languages)

... and why we didn't choose one, yet

- principal decision (CSCM or DSCM)
- all candidates were lacking in certain areas
  - Mercurial
    - no real support for branches
    - slow
  - Git
    - `ls /usr/bin/git-* | wc -l` => 124, docs are cryptic in parts
    - Windows support is only second class
    - very large repositories need to be broken up
  - SubVersion
    - is fast only via svn protocol
    - manual merge tracking

# Meanwhile: easing the pain with PCVSLib

- **cwsresync: by far the most painful tool**
  - for old big CWSs the time needed for rebase was many hours, possibly even days!
- **the reason**
  - called the cvs client several times per file
  - cvs command line client is quite inflexible
  - lousy error reporting
- **solution: use PCVSLib**
  - perl implementation of a CVS client library
  - right on top of the CVS protocol

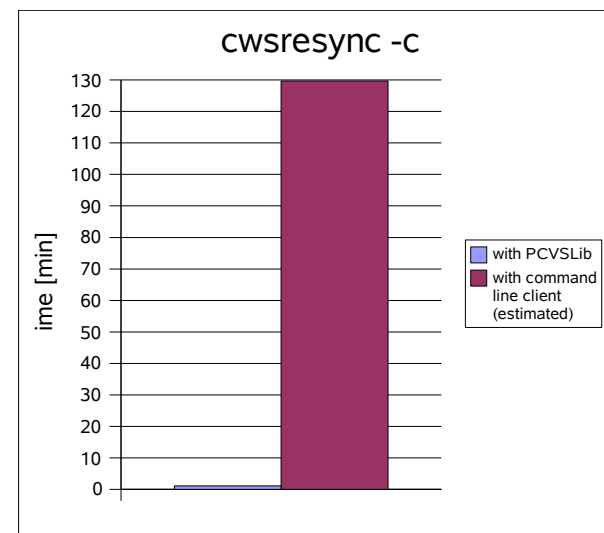
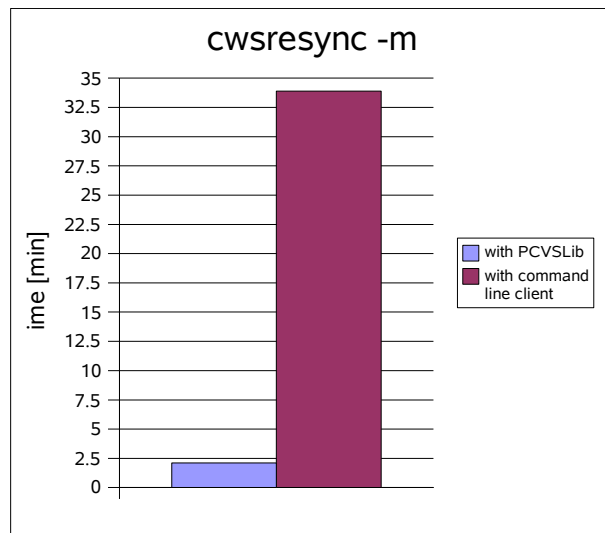
# PCVSLib

- native perl
- lives for now in “solenv”, later maybe CPAN
  - (if I ever get around to proper documenting it)
- uses an event based model
  - got inspirations from netbeans javacvs library
  - allows client to inform about progress of operation
  - far better error checking
  - very fine grained control
  - allows multiple operations with only one connection

# PCVSLib applied to cwsresync:

## Rebasing an old CWS

- typical scenario: 778 changes betw. milestones



```
use PCVSLib;
```

```
$root = PCVSLib::Root->new("... root ...");  
$credentials = PCVSLib->Credentials->new();  
$passwd = $credentials->get_password($root);  
$conn = PCVSLib::Connection->new($root, $passwd);  
if ( !$conn->open() ) { ... die horribly ... };  
$client = PCVSLib::Client->new($conn);
```

```
$eh = PCVSLib::EventHandler->new();  
$listener = SimpleListener->new()  
$eh->add_listener($listener);
```

```
$command = PCVSLib::CheckoutCommand->new($eh);  
$command->file_list(['... files ...']);  
$client->execute_command($command);  
$con->close(); $eh->remove_listener($listener);
```

```
if ( !$listener->is_success() ) { .... }
```

```

package SimpleListener;

sub new {
    ... $self->{is_success_} = 0; ...
}

sub is_success {
    my $self = shift; return $self->{is_success_};
}

sub notify {
    my ($self, $event) = @_;
    if ( $event->isa(PCVSLib::TerminateEvent) ) {
        $self->{is_success_}=$event->is_success();
    }
    if ( $event->isa(PCVSLib::MessageEvent) ) {
        print $event->get_message() . '\n';
    }
}
}

```