Configuration Management & Continuous Integration

Lutz Küderli / Clemens Lanthaler





- Many people working on project
- How to keep track of changes?
- How do you eliminate bugs as early as possible?

- Example: 100 modules, 10 developers, 20 versions of every module – every developer changes 1 module a day
- Since yesterday, programmers changed 10 modules if something doesn't work – what am I going to do?



Examples

- Identification & Tracking
 - "Yesterday, this still worked!"
 - "I already fixed this last week!"
 - " This is obvious improvement. Has this already been tested?"
- Version selection
 - " Has everything been compiled?"
 - " How do I exclude this flawed change?"
 - "How to configure a test with my changes only?"



- Identification of configuration items
- Change control
- Status accounting
- Auditing
- Build management
- Process management



- Continuous Integration
- CVS
- Communication

ARENA Continuous Integration I

- Benefits
 - Bugs appear earlier in the SE process
 - Faster to track the bug
 - Saves lots of time when it comes to integrating the whole product
 - Does NOT catch all integration bugs
- The More Often The Better
 - Build at least once a day/night
 - Example: Microsoft does nightly builds for projects with >30 mill LOC
 - Absolutely necessary to automate build process

ARENA Continuous Integration II

- What is a successful build?
 - All the latest sources are checked out of the configuration management system
 - Every file is compiled from scratch
 - The resulting object files (Java classes in our case) and linked and deployed for execution (put into jars).
 - The system is started and suite of tests is run against the system.
 - If all of these steps execute without error or human intervention and every test passes, then we have a successful build

ARENA Continuous Integration III

- Single Source Point
 - That's why we use CVS
- Automated Build Scripts
 - With lots of files, it's not going to be possible to compile everything by hand
 - We are going to use ANT
- Checking In
 - IMPORTANT: Check your code in early!
- Regression Testing



- Concurrent Versions System
- Enables groups of persons to work simultaneously on source code
- Central *repository* holds all versions of the source code, dividing it into *modules*
- You can *check out* code from the repository, edit it and *commit* it again
- Possible to fetch every earlier version from the repository
- Detects and marks conflicts



CVS enviroment variables

CVSROOT

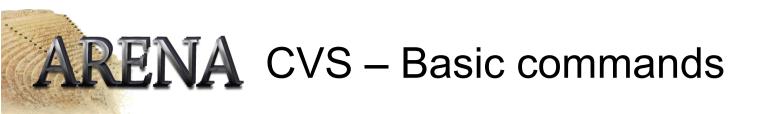
(setenv CVSROOT :ext:<login>@cvsbruegge.in.tum.de:/cvs/arena)

CVSEDITOR

(setenv CVSEDITOR emacs) specify the editor for comments (default is vi)

CVS_RSH

must point to ssh (setenv CVS_RSH ssh)



- cvs checkout <module> (Cvs CO)
 - Creates a local copy of the repository
- cvs update
 - Updates you local copy from the repository
- cvs commit
 - Checks in your changed files
- cvs add cvs remove
 - Adds or removes files from the repository
- cvs log
 - Shows the comments made by the persons who committed it
- Many many more ... man cvs is your friend



• To get a copy of the repository you must use the following command:

```
cvs checkout [-D <Date>] <module>
```

```
(example: cvs checkout -D yesterday . )
```



- Get a working copy of the repository cvs checkout <module> (e.g. cvs checkout .)
- Create the new file
- Checkin the new file cvs add [-m <description>] <filename>
- cvs commit [-m "Early version"] <filename>



- Get a working copy of the repository cvs checkout <module>
- Delete the file(s)
- Call: cvs remove to mark the files for removing
- Call: cvs commit to remove the files from the repository



- Get a working copy of the repository cvs checkout .
- Rename the file (mv <oldfile> <newfile>)
- Call: cvs remove <oldfile>
- Call: cvs add <newfile>
- Call: cvs commit <oldfile> <newfile>



CVS Example

- Need a running cvs here ③
- WebCVS access: http://cvsbruegge.in.tum.de/cgi-bin/cvsweb.cgi



- Think before you check in!
- Always make sure your code is compiling properly before checking it in
- Add useful comments!
- Avoid committing binaries (*.class); sometimes it is necessary, though
- Try to resolve conflicts as soon as possible
- Remember you can always "roll back" your version



Thank you for your attention !