XML Praktikum

Session 1 - 18.10.2001

Dan Olteanu, Tim Furche, Tobias Kiesling

xmlprakt@informatik.uni-muenchen.de
Contents

1 What is Concurrent Versions System for?  3

2 CVS for XML Praktikum  10

3 What else to know  13
What is Concurrent Versions System for?

- maintains a history of a source tree in terms of a series of changes
- stamps each change with the corresponding time, user name and motivation
- allows concurrent work of several users within the same project
- CVS vs. RCS: RCS provides only a **locking** mechanism, i.e. only one user can work on a file at one time
- CVS does **not** understand the semantics of your program.

Hence, **Concurrent Versions System** answer questions like:

- *Who* made a given change?
- *When* did they make it?
- *Why* did they make it?
- *What* other changes did they make at the same time?
For the first time..

Setting your repository

setting the CVSROOT environment variable:

setenv CVSROOT <repository_path> for csh-like shell

Checking out a working directory

• each user has to work within a directory created by CVS, i.e. a local working copy of the current status of the project

• use cvs checkout <module> for getting a directory tree from CVS
  “Check out the source tree called module from the repository specified by the CVSROOT environment variable”

• under local <module> directory, there is a subdirectory CVS with management information
Modifying files

- the changes of each user are not automatically visible to the others
- each user should commit his modifications to the repository for making them available to the others
- before changing, it is reasonable for local sources to be in sync with any changes committed by the others
- `cvs update` merges the central sources with the local ones from an user, i.e. marks them as *updated* or *modified*
- after updating, one can commit the modified source files: `cvs commit <file_or_dir>`
- prompt for a log message describing the change
Examining changes

- `cvs log <file_or_dir>`,
  to see the changes on `<file_or_dir>`

- `cvs diff -c -r 1.6 -r 1.7 <file_or_dir>`,
  to see the changes one committed as revision 1.7
Adding & deleting files

- file creation or deletion treated like other changes by recording such events in the history of files
- CVS does not automatically take over the newly created files
  ```
cvs add <file_or_dir>
cvs rm <file_or_dir>
  ```
- simple deletion of files, i.e. without deleting them from the repository, can be seen as undo on all your changes since last update
- strategy for renaming a file:
  - rename the file in OS
  - remove the old file from repository
  - add the new file to repository
- do not forget to commit these changes!
Handling conflicts

- conflict: more than one user modified simultaneously the same lines in a file

- CVS cannot resolve the conflicts, but can help you in solving them, by
  - marking the files with conflicts
  - marking the conflict regions in the files

- usually, conflicts are due to
  - attempt to address at the same time the same problem
  - lack of communication between developers
  - disagreement about the design of the program
CVS resources

- CVS links
  - CVS homepage
    http://www.cvshome.org
  - CVS Manual on Web
  - Projektverwaltung mit CVS
    http://web.informatik.uni-bonn.de/II/ag-anlauf/cvs/cvs-handout.html
  - The CVS Book
    http://cvsbook.red-bean.com
- CVS in XEmacs
CVS for XML Praktikum

- go in the directory allocated for your team, e.g.
  \texttt{cd /home/proj/xmlprakt/teams/xmlvoice}

- checkout for the first time, e.g.
  \texttt{cvs checkout ucis}

- modify as you like the local copy

- update your copy with the changes of the others, e.g.
  \texttt{cvs update}

- commit your changes, e.g.
  \texttt{cvs commit <files_you_change>}

- resolve the possible conflicts
CVS repositories in the tutorial

- CVS repositories in /home/proj/xmlprakt/cvs
- include this path by setting environment variable CVSROOT or provide it to cvs command with option '-d' on checkout
- the following repositories exist:

<table>
<thead>
<tr>
<th>doc</th>
<th>documentation for the system; every piece of documentation you create should be added to this repository</th>
</tr>
</thead>
<tbody>
<tr>
<td>lib</td>
<td>system-specific library, e.g. Java classes used for the system</td>
</tr>
<tr>
<td>ucis</td>
<td>university course information system; should only contain the files that need to be accessed by the user</td>
</tr>
</tbody>
</table>
What else to know

• Testing:
  – Access UCIS with a standard web browser by using the URLs below
  – Start by using the path lehreinheiten/lehreinheiten.xml
  – To access APRIL data, use debug script from bin directory

• URLs for UCIS:
  – Central instance: http://marutea.pms.informatik.uni-muenchen.de/ucis
  – Instances for teams:
    http://marutea.pms.informatik.uni-muenchen.de/xml(april|db|path|voice)

• Restarting the tomcat server:
  – Log on to marutea.pms.informatik.uni-muenchen.de via ssh
  – Shutdown: sudo /home/proj/xmlprakt/bin/shutdown.sh
  – Startup: sudo /home/proj/xmlprakt/bin/startup.sh

• Mailing lists:
  xml(all|april|db|path|prakt|voice)@pms.informatik.uni-muenchen.de