SEPM/Linux Tasks Winter Term 2009

SE/Linux Team*

December 28, 2009

1 Task 4

1.1 Focus

This task focuses on implementing and testing the design and project plan from the previous task with your team.

1.2 Mission

Release two working versions of an EHR provider node.

1.3 Tasks

Your team has designed and planned the implementation of a provider node. Now execute the design and plan you made. You will have to release two working versions of your server and present them to your tutor. It is important that you work in accordance with your plan. If you need to deviate from the plan be sure to document any changes and keep your design and plan up-to-date.

1.3.1 Reading

- Read the Boost test library documentation at [1].
- Read the Bitten documentation at [2].
- Read the Doxygen documentation at [4].
- Read the libpqxx documentation at [6].
- Read the libxml++ documentation at [7].

*se-linux@inso.tuwien.ac.at

1.3.2 Designing and coding

- Use your group's subversion repository.
- We have created a PostgreSQL [5] database for your group. Access it using libpqxx [6].
- Provide an SQL script to initialize your database.
- Use libxml++ [7] for XML processing.
- Close tasks (tickets) in Trac as soon as they are implemented.
- Use Trac as a communication (or at least management) medium in your team.
- Keep the design document up to date if you change the design.
- If you encounter bugs while working:
  - Create a ticket in Trac.
  - Assign it to the responsible team member.
  - The responsible team member has to accept, fix and close it.

1.3.3 Unit tests

Use the Boost test library [1] to write unit tests for your classes and functions. As a rule of thumb, the unit tests should cover all your application classes, i.e. every source line of your application classes should be executed by at least one unit test.

- It is important for your classes to be testable in an automatic way. This can be achieved by making them not depend on human interaction and decoupling them from their environment.
- Provide a script that runs all unit tests of your project, or integrate them with Automake.
- Automake provides support for test suites [3] via a `check` target that builds all tests, executes them and displays the results.

The exercise tarball contains examples for writing unit tests.

1.3.4 Continuous integration

Use the Bitten [2] Trac plugin to set up automatic builds of your subversion repository on every commit. For this, you need to run `bitten-slave` in a `screen(1)` session on both target platforms. An automatic build should:

- Check out the latest revision of your source tree

- Configure your source tree
- Build the software to see whether the build has not been broken by the latest commits
- Build all unit tests
- Run all unit tests and report the test results
- Determine the tests' code coverage using `gcov(1)`[1] and report it.

The exercise tarball contains an example for this.

1.3.5 Documentation

Your source code has to be documented.

- Document your C++ classes by inserting Doxygen-conform comments into your header files.
- Write a Doxygen configuration file.
- Use Doxygen to generate HTML documentation.

1.4 Deliverables

- Tag the first release in your subversion repository in `/submission/release_1`.
- Tag the second release in your subversion repository in `/submission/release_2`.
- Pre-create Doxygen documentation in HTML format for every release before you meet your tutor, so you can present it easily.
- Update the design document in your Trac instance if necessary.
- Close tickets from task 3 in your Trac instance.

[1] At the time of this writing, compiling with gcov support is broken on the NetBSD shell server, so perform coverage analysis on the GNU/Linux server only.

References

[1] Dawes, Beman ; Abrahams, David ; Rivera, Rene ; Rozental, Gennadiy: *Boost C++ Libraries - Boost Test Library*. http://www.boost.org/doc/libs/1_35_0/libs/test/doc/index.html. Version: 2008. – [Online; accessed 2008-05-25]

[2] Edgewall Software: *Bitten*. http://bitten.edgewall.org/. Version: 2008. – [Online; accessed 2008-05-25]

[3] Free Software Foundation: *Tests - automake*. http://www.gnu.org/software/automake/manual/html_node/Tests.html#Tests. Version: 2008. – [Online; accessed 2008-05-25]

[4] Heesch, Dimitri van: *Doxygen*. http://www.stack.nl/~dimitri/doxygen/. Version: 2007. – [Online; accessed 2007-03-18]

[5] PostgreSQL Global Development Group: *PostgreSQL*. http://www.postgresql.org/. Version: 2008. – [Online; accessed 2008-05-11]

[6] The libpqxx Development Team: *libpqxx*. http://pqxx.org/development/libpqxx/. Version: 2008. – [Online; accessed 2008-05-11]

[7] The libxml++ Development Team: *libxml++*. http://libxmlplusplus.sourceforge.net/. Version: 2007. – [Online; accessed 2007-02-26]