SIP- The Software Improvement Process

Tools and Rules to Encourage Quality

Katarina Sigerud, CERN
13th October, ICALEPCS 2011
Objectives

 Introduce quality improvement as an integral part of the everyday development work
 Leverage tools to automate the process as much as possible
 Establish guidelines and metrics to measure progress
Quality in the development cycle

For each stage

- Agreed mandatory activities and deliverables
- Tools identified and integrated with IDE (Eclipse), giving immediate feedback
Design reviews

- To verify the soundness of design in an early stage
- To identify overlapping functionality
- At the level of sub-components, main classes and design patterns
- UML: class and sequence diagrams
Code reviews

Goal:
- Identify defects
- Ensure maintainable code
- Verify conventions are followed

Static code analysis tools identifies common mistakes and bug patterns:
- FindBugs
- Checkstyle
- Eclipse warnings

Person-to-person time consuming
- Only for critical code
- Mentoring of junior developers

Light-weight person-to-person code reviews with FishEye + Crucible
private int parseInt(String numberString) {
    return Integer.parseInt(numberString == null || numberString.equalsIgnoreCase(""));
}

private TopFlopMetrics prepareMetricsFromTest() throws ParserConfigurationException, SAXException, TopFlopMetrics metrics = new TopFlopMetrics();

    File bambooBuildDir = new File(
"C:\\Users\\ksigerud\\eclipse\\projects\\sandbox\\sip-top-flop\\src\\test-r"
    File[] listFiles = bambooBuildDir.listFiles();
    for (int i = 0; i < listFiles.length; i++) {
        File cloverFile = listFiles[i];
        if (!cloverFile.isDirectory() && cloverFile.getName().contains("-coverage")) {
            String projectName = cloverFile.getName().substring(0, cloverFile.getName().length() - "-coverage).length());
            metrics.addProjectMetrics(parseCoverageFile(cloverFile, projectName));
        }
    }

    return metrics;
}
A list of 'bugs'
A list of bugs

- Code contains a hard coded reference to an absolute pathname (2)
- Hard coded reference to an absolute pathname
- Dead store to local variable (1)

The ‘bug’ line indicated
A list of 'bugs' indicated:

- Code contains a hard coded reference to an absolute pathname (2)
- Hard coded reference to an absolute pathname
- Hard coded reference to an absolute pathname
- Dead store to local variable (1)

The 'bug' line indicated:

```java
private TopFlopMetrics prepareMetricsFromTest() throws ParserConfigurationException, SAXException, TransformerException {
    TopFlopMetrics metrics = new TopFlopMetrics();

    File bambooBuildDir = new File("C:\\Users\\ksigerud\\eclipse\\projects\\sandbox\\sip-top-flop\\src\\test-r");
    File[] listFiles = bambooBuildDir.listFiles();
    for (int i = 0; i < listFiles.length; i++) {
        File cloverFile = listFiles[i];
        if (!cloverFile.isDirectory() && cloverFile.getName().contains("-coverage")) {
            String projectName = cloverFile.getName().substring(0, cloverFile.getName().length());
            metrics.addProjectMetrics(parseCoverageFile(cloverFile, projectName));
        }
    }
}
```

'Bug' explained:

```
Bug: Hard coded reference to an absolute pathname
Pattern id: DMI_HARDCODED_ABSOLUTE_FILENAMe, type: DMI, category: STYLE

This code constructs a File object using a hard coded to an absolute pathname (e.g., new File("
/home/danny/workspace/j2ee/src/share/com/sun/enterprise/deployment"));
```
## Top/flop list review

Closed on 09 Oct

### Details

<table>
<thead>
<tr>
<th>Participant</th>
<th>Role</th>
<th>Time Spent</th>
<th>Comments</th>
<th>Latest Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Katarina Sigerud</td>
<td>Author</td>
<td>3m</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pablo Pera Mira</td>
<td>Reviewer - 100% complete</td>
<td>18m</td>
<td>2</td>
<td>I suggest you use the builder pattern here this long...</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>21m</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Files: 1

### Objectives

Please review this class using the Eclipse Atlassian connector plugin in view of trying it out.

### Summary

Thank you for your help! It seems that the Eclipse Atlassian connector still needs some work...

### General Comments

**Pablo Pera Mira** says:

After a couple of hours fighting playing with several installations of Eclipse, I managed to get it all configured in my laptop... Installing/uninstalling, and especially upgrading plugins is pretty painful!

It's nice to navigate the files from Eclipse, it lets you check all changes way faster. It's not essential, though, especially for small reviews.
Files being reviewed

<table>
<thead>
<tr>
<th>Participant</th>
<th>Role</th>
<th>Time Spent</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Katarina Sigerud</td>
<td>Author</td>
<td>3m</td>
<td></td>
</tr>
<tr>
<td>Pablo Pera Mira</td>
<td>Reviewer - 100% complete</td>
<td>18m</td>
<td>2</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>21m</td>
<td>2</td>
</tr>
</tbody>
</table>

Summary

Thank you for your help! It seems that the Eclipse Atlassian connector still needs some work...

General Comments

Pablo Pera Mira says:

After a couple of hours fighting playing with several installations of Eclipse, I managed to get it all configured in my laptop... Installing/uninstalling, and especially upgrading plugins is pretty painful!

It's nice to navigate the files from Eclipse, it lets you check all changes way faster. It's not essential, though, especially for small reviews.
Files being reviewed

Author and reviewers
Comments inline
Testing

- General agreement on benefits of unit tests
- To increase level of testing, unit tests mandatory deliverable of project
  - <30% test coverage for non-trivial classes, measured with Clover
- To discover inter-project issues early, continuous builds with Bamboo
  - Compiles and runs unit tests
```java
String[] result = new String[originalValues.length];
for (int i = 0; i < originalValues.length; i++) {
    int n = this.annotations.indexOf(originalValues[i]);
    if (n >= 0) {
        result[i] = (String) this.targets.get(n);
    }
}
return result;

/** Base class for all prepared statement callbacks which execute queries in a batch mode. Derived classes are
 * required to compute expected batch size and return actual updated row count.
 * @author Wojciech Slivinski
 */
public static abstract class BatchUpdateCallback implements PreparedStatementCallback {

    // Define fixed batch size, Oracle recommends values in a range (5,30) -->
    // Oracle JDBC Developers Guide and Reference
    private static final int BATCH_SIZE = 30;

    public final Object doInPreparedStatement(PreparedStatement ps) throws SQLException, DataAccessException {
        final int dataSize = computeDataEntriesCount();
        if (dataSize == 0) {
            // nothing to do, return
            return new Integer(0);
        }

        final OraclePreparedStatement ops = DaoHelper.convertToOraclePreparedStatement(ps);
        ops.setExecuteBatch(BATCH_SIZE);
        // call derived abstract method
        int updatedRowCount = doInPreparedStatementImpl(ops);
        // final flush of batch buffer
        int batchRowCount = ops.sendBatch();
        if (dataSize != updatedRowCount + batchRowCount) {
            if (LOGGER.isEnabled()) {
                LOGGER.info("Committed data entries size [" + dataSize + "] is different from updated row count [" + (updatedRowCount + batchRowCount) + "]");
            }
            return new Integer(updatedRowCount + batchRowCount);
        }

        protected abstract int computeDataEntriesCount();

        protected abstract int doInPreparedStatementImpl(OraclePreparedStatement ops) throws SQLException,
        DataAccessException;
```
Red = not covered

Green = covered
Triggered by changes in a dependency.
Triggered by changes in a dependency

accsoft-commons-util

accsoft-commons-core

accsoft-commons-concentration

accsoft-commons-util

accsoft-commons-util

accsoft-commons-core

accsoft-commons-concentration
Triggered by changes in a dependency

- accsoft-commons-core
- accsoft-commons-util
- accsoft-commons-concentration
- accsoft-commons-io

Top 10 Risk Projects (high complexity, low coverage):
- pm-converters
- pm-powering-server
- pm-server
- pm-viewers
- inca-common-utils
- jpcp-directory-client
- accsoft-commons-cache
- pm-db-journals
- pm-domain
- jpcp-ext-cmwrda

15 Best Covered Projects:
- 91.3% pm-converters
- 71.7% pm-core
- 63.4% pm-domain
- 60.2% pm-server
- 57.4% jpcp-ext-cmwrda
- 47.9% jpcp-ext-remote
- 43.3% jpcp-value
- 43% pm-viewers
- 40.8% accsoft-commons-cache
- 40.2% pm-powering-server
- 37.7% jpcp
- 37.7% inca-common-utils

Top 15 Complex Projects (ordered by complexity):
1. 37.7% jpcp
2. 43.3% jpcp-value
3. 71.7% pm-core
4. 21% inca-server-core
5. 24.7% accsoft-commons-utils
6. 47.9% jpcp-ext-remote
7. 57.4% jpcp-ext-cmwrda
8. 63.4% pm-domain
9. 20% inca-server-valcomp
10. 40.8% accsoft-commons-cache
11. 36.9% pm-db-journals
12. 37.7% inca-common-utils
13. 60.2% pm-server

15 Least Covered Projects:
15. 11% jpcp-directory-client
20. 21% inca-server-valcomp
21% inca-server-core
24.7% accsoft-commons-utils
36.9% pm-db-journals
37.7% inca-common-utils
37.7% jpcp
40.2% pm-powering-server
40.8% accsoft-commons-cache
43% pm-viewers
43.3% jpcp-value
47.9% jpcp-ext-remote
Top/flop list generated (Work in progress)

pm-converters
pm-powering-server
pm-server
pm-viewers
inca-common-utils
japc-directory-client
accsoft-commons-cache
pm-db-journals
pm-domain
japc

15 Best Covered Projects

<table>
<thead>
<tr>
<th>Project</th>
<th>Coverage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>pm-converters</td>
<td>91.3%</td>
</tr>
<tr>
<td>pm-core</td>
<td>71.7%</td>
</tr>
<tr>
<td>pm-domain</td>
<td>63.4%</td>
</tr>
<tr>
<td>pm-server</td>
<td>60.2%</td>
</tr>
<tr>
<td>pm-viewers</td>
<td>57.4%</td>
</tr>
<tr>
<td>japc-ext-cmwrda</td>
<td>47.9%</td>
</tr>
<tr>
<td>japc-ext-remote</td>
<td>47.9%</td>
</tr>
<tr>
<td>japc-value</td>
<td>43.3%</td>
</tr>
<tr>
<td>pm-viewers</td>
<td>43%</td>
</tr>
<tr>
<td>pm-powering-server</td>
<td>40.8%</td>
</tr>
<tr>
<td>pm-domain</td>
<td>37.7%</td>
</tr>
<tr>
<td>pm-converters</td>
<td>37.7%</td>
</tr>
</tbody>
</table>

15 Least Covered Projects

<table>
<thead>
<tr>
<th>Project</th>
<th>Coverage (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>japc-directory-client</td>
<td>15.1%</td>
</tr>
<tr>
<td>inca-server-valcomp</td>
<td>20%</td>
</tr>
<tr>
<td>inca-server-core</td>
<td>21%</td>
</tr>
<tr>
<td>accsoft-commons-util</td>
<td>24.7%</td>
</tr>
<tr>
<td>pm-db-journals</td>
<td>24.7%</td>
</tr>
<tr>
<td>pm-converters</td>
<td>36.9%</td>
</tr>
<tr>
<td>pm-viewers</td>
<td>37.7%</td>
</tr>
<tr>
<td>pm-powering-server</td>
<td>37.7%</td>
</tr>
<tr>
<td>pm-viewers</td>
<td>37.7%</td>
</tr>
<tr>
<td>inca-common-utils</td>
<td>37.7%</td>
</tr>
<tr>
<td>japc-ext-cmwrda</td>
<td>40.2%</td>
</tr>
<tr>
<td>pm-db-journals</td>
<td>40.8%</td>
</tr>
<tr>
<td>pm-converters</td>
<td>43%</td>
</tr>
</tbody>
</table>
Triggered by changes in a dependency:

- `accsoft-commons-core`
- `accsoft-commons-util`
- `accsoft-commons-concentration`
- `accsoft-commons-io`

Top/flop list generated (Work in progress):

- `pm-converters`
- `pm-powering-server`
- `pm-server`

15 Least Covered Projects:

<table>
<thead>
<tr>
<th>Project</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>japc-directory-client</td>
<td>15.1%</td>
</tr>
<tr>
<td>inca-server-valcomp</td>
<td>20%</td>
</tr>
<tr>
<td>inca-server-core</td>
<td>21%</td>
</tr>
<tr>
<td>accsoft-commons-util</td>
<td>24.7%</td>
</tr>
<tr>
<td>pm-db-journals</td>
<td>36.9%</td>
</tr>
<tr>
<td>inca-common-utils</td>
<td>37.7%</td>
</tr>
<tr>
<td>pm-powering-server</td>
<td>40.2%</td>
</tr>
<tr>
<td>accsoft-commons-cache</td>
<td>40.8%</td>
</tr>
<tr>
<td>pm-viewers</td>
<td>43%</td>
</tr>
<tr>
<td>japc-value</td>
<td>43.3%</td>
</tr>
<tr>
<td>japc-ext-remote</td>
<td>47.9%</td>
</tr>
</tbody>
</table>

Top 15 Complex Projects (ordered):

<table>
<thead>
<tr>
<th>Project</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>japc</td>
<td>37.7%</td>
</tr>
<tr>
<td>japc-value</td>
<td>43.3%</td>
</tr>
<tr>
<td>pm-powering-server</td>
<td>43%</td>
</tr>
<tr>
<td>pm-converters</td>
<td>47.9%</td>
</tr>
<tr>
<td>japc-ext-remote</td>
<td>21%</td>
</tr>
<tr>
<td>inca-server-core</td>
<td>24.7%</td>
</tr>
<tr>
<td>accsoft-commons-util</td>
<td>36.9%</td>
</tr>
<tr>
<td>pm-db-journals</td>
<td>40.2%</td>
</tr>
<tr>
<td>inca-common-utils</td>
<td>43%</td>
</tr>
<tr>
<td>japc</td>
<td>43.3%</td>
</tr>
<tr>
<td>pm-viewers</td>
<td>47.9%</td>
</tr>
<tr>
<td>inca-server-valcomp</td>
<td>21%</td>
</tr>
<tr>
<td>accsoft-commons-cache</td>
<td>40.8%</td>
</tr>
<tr>
<td>pm-powering-server</td>
<td>43%</td>
</tr>
<tr>
<td>inca-common-utils</td>
<td>47.9%</td>
</tr>
</tbody>
</table>
Top/flop list generated (Work in progress)

- pm-converters
- pm-powering-server
- pm-server

15 Best Covered Projects:

- 91.3% pm-converters
- 71.7% pm-core
- 63.4% pm-domain
- 60.2% pm-server
- 57.4% jpc-ext-cmwrda
- 47.9% jpc-ext-remote
- 43.3% jpc-value
- 43% pm-viewers
- 40.8% accsoft-commons-cache
- 40.2% pm-powering-server
- 37.7% jpc
- 37.7% inca-common-utils
Challenges
Agree on standards and configurations

- Think of and organize us as one big team, not many small ones
- The code belongs to the group, not to a project or an individual developer
- The guidelines and standards established and agreed by everyone
Encourage developers to invest time on quality

- Quality-related tasks part of the yearly objectives for a project
- Progress measured against the level a project adheres to guidelines – top/flop lists
- Focus the effort where the effect is highest, rely on tools as much as possible
- SIP days – common days with a common goal
SIP an initiative to integrate quality improvement in the development cycle

A number of guidelines/standards and supporting tools have been agreed

Developer motivation through
  - Immediate feedback when developing
  - Official tracking of progress and top/flop lists

Increased awareness of benefits, application in individual projects and confidence when introducing changes