1. Why does the described automated testing infrastructure not catch the defect?

In Junit the attribute "haltonfailure=false" we used in the build.xml file means that, even if tests fail, the build would continue and exit with an 0 code since the build itself is completed successfully. If we want to get non 0 code when there's a test failure, we should assign true to haltonfailure.

2. How could the developers improve the testing infrastructure (either manual or automated) to immediately notice test failures in the future? One way to fix the issue is, like the solution above, set the value of haltonfailure attribute to true. Another way to solve it, although it might require some more labor, is to improve the notification mechanism. Send email with log messages included even if all tests passed, so people might notice if they check their email inbox regularly.

3. For git bisect, how many commits exist between v1.0.0 and the HEAD revisions (including v1.0.0 and HEAD)? What command(s) did you use to determine the number?

There are 37 commits between v1.0.0 and HEAD. We use command `git log v1.0.0^...HEAD --oneline | wc -l`.

4. Based on the git bisect results, which commit (commit hash and log message) introduced the defect? How did you independently verify (meaning another approach other than git bisect) that this commit indeed introduced the defect?

f7f175dc8bc5b3693f99b2f8e799b51c0d0d9b9f is the first bad commit

```
git checkout f7f175dc8bc5b3693f99b2f8e799b51c0d0d9b9f^
ant clean test

// the version before the claimed first bad commit should pass all the
tests
```