Software Engineering - Exercise Sheet 10

Martin Mehlmann
(mehlmann@informatik.uni-freiburg.de)

July 7, 2009

Exercise 1.1

1. We observe a *failure* which is a null pointer exception in this case.
2. The declaration of pi as 31.4 is a *code smell* i.e. likely to be a *defect*. If we can relate the failure of the single testcase to pi, we know that the declaration of pi as 31.4 is a *defect*. Note that, if we can't relate the failure to the definition of pi, this does not mean that the definition of pi is correct.
3. The statement talks about a value of a variable z and is therefore related to the program state at some point during execution. If the value of the variable violates the specification, we have an *infection* of the program state which may *propagate* via other variables which depend on z.
4. We remove a bug or in other words, we fix a *defect* which was distributed over three files in this case.
5. We observe another time than the expected one. Therefore we observe a *failure* if the clock is actually intended to show the local time.

Exercise 1.2

1. No, a defect only results in an infection of the program state if it is actually executed by one of the test cases in your test suite
2. No, either the infection is not visible as a program failure or the infection is "healed" during execution again.
3. Yes, consider very rare cases where cosmic radiation or other rare events may lead to a bit-flip. Another possibility where an infection may occur without a defect in the program code is invalid external data, e.g. read from a file.

Exercise 3

- If we prove a program correct using another program (the proofer) we assume that the proofer works correctly which may not be the case. Therefore the program may still contain undetected defects because the proofer has.
- If we prove a program we always prove it with respect to some specification. However, the specification itself may already be incorrect, incomplete or unintended.
- Testing and Debugging can not only increase the confidence in the correctness of a program but also lead to a better overall understanding of the program code.

Exercise 4

```
import org.junit.*;
import static org.junit.Assert.*;
import java.net.URL;

public class URLTest extends Testcase {

    @Before
    public void setUp() {
        String s;
        s = "http://www.somehost.com/somepath.php?query=whatever";
        this.url = new Url(s);
    }

    @After
    public void tearDown() {
        this.url = null;
    }
```

Exercise 4

```
@Test
public void test_getProtocol() {
    assertEquals(this.url.getProtocol(), "http");
}

@Test
public void test_getHost() {
    assertEquals(this.url.getHost(), "www.somehost.com");
}

@Test
public void test_getPort() {
    assertEquals(this.url.getPort(), 80);
}
```

Exercise 4

```
    @Test
    public void test_getPath() {
        assertEquals(this.url.getPath(), "somepath.php");
    }

    @Test
    public void test_getQuery() {
        assertEquals(this.url.getQuery(), "query=whatever");
    }

    private URL url;
}
```