201300180 Data & Information – Test 1 (1.5 hours)

1 May 2014, 13:45–15:15

Solutions

Question 1

There are different good answers. The answer below is an example that scores the full 30 points (in fact slightly higher – functional correctness is not at all obvious from the lecture and the lab exercise, anyone who thinks of this gets a small bonus). Other answers need not be wrong. They score higher or lower, depending on the motivation you give why this is important. Motivations that are specific for the case (e.g. "User error protection is important, in order to diminish the risk that a call for help is missed") score better than generic motivations that would apply to any system (e.g. "User error protection is important, in order to make sure that the system is only used in the right manner")

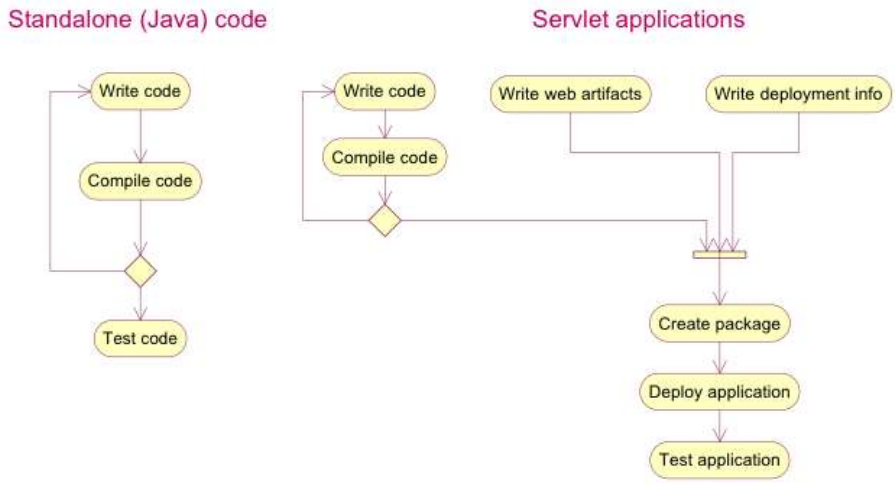
1. Functional correctness, because the number of false negatives should be (nearly) zero, the number of false positives should be low.
   (Note: If it is easy to specify functional requirements (user stories) with acceptance criteria that clearly distinguish correct and incorrect functionality, then functional correctness is a functional requirement rather than a quality requirement. This case is different, because it is impossible to implement the functionality 100 % correctly – 100% is not even aimed for, the necessity of false positives is taken into account. So the quality increases as their number decreases).
   Example: "The number of false positives should be lower than 10 % of the incoming calls."

2. Availability. Any time the system doesn't work at the client's end, someone has be sent over. And, more importantly, if the system in the TZC call centre doesn't work, calls for help cannot be answered.
   Example: "A downtime of the central system up to 15 minutes should occur less than once a year, otherwise any downtime should be less than 5 minutes."

3. Usability (or, more specific, Learnability or Operability). Perhaps you can shout "help" in different manners and dialects, and when it's installed it makes sense to test whether the client can use it.
   Example: "95 % of the clients can make a connection after a single test session of 10 minutes."

Question 2

a) The main difference is that while in the development of a standalone Java application after compilation the application is ready to be tested, with a Servlet-based web application it is also necessary to package all the artifacts (all the compiled code, web pages, style sheet files, images, etc.) and deploy them in a web container (application server). The process is thus more complex. The figure below shows this difference.



b) The deployment descriptor tells the servlet which URL corresponds to which servlet, and which Java class implements the servlet. The deployment descriptor allows the application server to determine where a HTTP request should be forwarded to (to which Servlet), based on its target URL.

c) The HTTPSession class allows the server to keep information about a certain client, the so called 'session tracking', circumventing in this way the stateless character of HTTP (HTTP does not keep information about former HTTP request-response exchanges). HTTPSession allows the server to keep a session with a client (browser) that lasts longer than a single HTTP message exchange.

When an HTTP request message arrives, the server can ask its related HTTPSession and inspect all the objects related to the session associated to this request. The server can also create a session, assign it to the HTTP request and assign attributes to the session, which can be inspected when a new HTTP request from the same browser arrives.

Question 3

a)
```
SELECT DISTINCT m.name, l1.language
FROM Movie m, Language l1, Language l2
WHERE l1.mid = m.mid
  AND l2.mid = m.mid
  AND l2.language = 'German'
  AND l1.language <> l2.language
```

b)
```
SELECT g.genre, MIN(m.rating) AS minimum, MAX(m.rating)
FROM Genre g, Movie m
WHERE g.mid = m.mid
GROUP BY g.genre
HAVING AVG(m.rating) >= 8
ORDER BY minimum
```

c)
```
SELECT DISTINCT m.name, m.year
FROM Movie m, Directs d, Acts a1
WHERE d.mid = m.mid
  AND d.pid = a1.pid
           -- i.e. the director of m is an actor
  AND NOT EXISTS (
           -- an actor (possibly a different one)
           -- who is a writer of m
    SELECT *
    FROM Acts a2, Writes w
    WHERE m.mid = w.mid
    AND w.pid = a2.pid )
```