Author: Santo Leonardo

Date: 2020-09-16

# Concrete Quality Attribute Scenarios

**Scenario 1 - retrieve common items in heavy load**

Identified as: **risk**

Explanation: the scenario foresees the environment in heavy load and the retrieval of other user's items, which are stored in a common database server.

Therefore, there is the risk that many concurrent requests on the same server cannot be satisfied in a reasonable time, impacting the requested response measure.

**Scenario 2 – create account**

Identified as: **risk, sensitivity point**

Explanation: new registrations are designed to be performed in the common database, and during normal operations they should be managed in the requested time.

Web server availability is a sensitivity point for many scenarios.

**Scenario 3 - login**

Identified as: **risk, trade-off**

Explanation: new logins are designed to be performed in the common database, and during normal operations they should be managed in the requested time.

It is a trade-off because for security reason the data are centrally stored (not locally), however this choice can impact the login time.

**Scenario 4 – validate empty filed**

Identified as: **risk**

Explanation: the input validation is done locally in the mobile app, therefore the response measure should be well addressed by the design of the app. The scenario is a risk because no quality attribute was selected to cover this scenario.

**Scenario 5 – add new item**

Identified as: **risk**

Explanation: adding a new item implies the storage in a common database, therefore depending on the response time of the web server on multiple requests.

**Scenario 6 – data change on server**

Identified as: **risk, sensitivity point**

Explanation: the web server has to update all the web and mobile apps, therefore this operation will depend on the number of active users, making it a risk. The artifact web server is in this case a sensitivity point.

**Scenario 6 – concurrent changes**

Identified as: **risk**

Explanation: the changes are made on the same record, one after the other one, therefore the latency of the second change is dependent on the processing of the previous change.

## Updated Utility Tree