Designing test cases

This project contains a lot of different parts to focus on. Examples are Uri testing, reading, writing, states, behavior of separate classes and the behavior as a whole. Not all functions should be tested when a user is installing this software, because several tests are designed to follow server behavior when shutting down or starting up while reading or writing amongst other things. Some of the edge cases might fail caused by race conditions. These cases might never be encountered under normal use and therefore not necessary to test.

What to test

Normal day to day tests

- Creating a database and collection.
- Using find to read.
- Using run-command to write, update and delete etc.
- Using run-command to get information.

Behavior tests

- Client behavior accessing servers defined by uri.
- Driver behavior when a server goes down, starts up or changes state.
- States where a driver can be in. These are held in the Client and Server objects.
- Behavior tests are done against servers of different versions.

Other tests

- Independent class tests like on Uri and logging.
- Replica server tests.
- Accounting tests
- Authentication tests.

When and where to test

The day to day tests are the tests placed in directory **./t**. The other tests are found in directories **./xt**.

User install from ecosystem

- Only day to day tests are done.

On my system, a Fedora 24+, 4-core, 8 threads

- Mostly day to day tests are done.
- From time to time other tests.

Travis-CI and Appveyor

Travis-CI and Appveyor (taken up later) are test systems where the software is installed and tests once a git push is executed. Travis is on a Ubuntu linux and the Appveyor is for windows systems.

- Day to day tests are done. This will set the outcome of the whole test.
- A select set of other tests which will change depending on the history (of failures). This will not influence the test result when one of the tests fail. Its purpose is mainly to see what happens in a driver.

Day to day tests

At most two servers are started for different versions for 2.6.* and 3.*

- Simple class tests of classes not (too much) depending on each other like Uri, Logging etc.
- Simple operations tests like create database and collection, read and write documents, substitutions, deletes and drop collections or databases.
- More complex operations such as index juggling, mapping and information gathering.

The tests

Test server table. In this table, the key name is saying something about the server used in the tests. This key is mentioned below in the test explanations below. There are also key combinations such as s1/authenticate which means that the particular server is started with additional options, in this case authentication is turned on.

Config key	Server version	Server type
s1	3.*	mongod

Simple cases

- Uri string tests in **xt/075-uri.t**. Can be placed in day to day test set. No server needed.
  - ✓ Server names
  - ✓ Uri key value options
  - ✓ Default option settings
  - ✓ Username and password
  - ✓ Reading any of the stored values
  - ✓ Failure testing on faulty uri strings

The MongoDB Client, Server, Monitor and Socket classes

These classes can not be tested separately because of their dependency on each other so we must create these tests in such a way that all can be tested thoroughly. Tests are not for day to dat tests.

- Client object interaction tests in **t/110-client.t**.
  - Unknown server which fails DNS lookup.
    - ✓ server can not be selected
    - ✓ server state is SS-Unknown
    - ✓ topology is TT-Unknown
  - Down server, no connection.
    - ✓ server can not be selected
    - ✓ server state is SS-Unknown
    - ✓ topology is TT-Unknown
  - Standalone server, not in replicaset. Use config s1.
    - ✓ server can be selected

        - ☑ server state is SS-Standalone
        - ☑ topology is TT-Single
    - Two standalone servers. Use config s1 and s2.
        - ☑ server can not be selected
        - ☑ both servers have state SS-Standalone
        - ☑ topology is TT-Unknown
- Client/server interaction tests in **t/111-client.t**.
    - Standalone server brought down and revived, Client object must follow. Use config s1.
        - ☑ current status and topology tested
        - ☑ shutdown server and restart
        - ☑ restarted server status and topology tested
    - Shutdown/restart server while inserting records. Use config s1.
        - ☑ start inserting records in a thread
        - ☑ shutdown/restart server
        - ☑ wait for recovery and resume inserting
- Client authentication tests in **t/112-client.t**.
    - Account preparation using config s1
        - ☑ insert a new user
    - Restart to authenticate using config s1/authenticate
        - ☑ authenticate using SCRAM-SHA1
        - ☑ insert records in users database is ok
        - ☑ insert records in other database fails

Tested	Test Filename	Test Purpose
x	610-repl-start	Replicaset server in pre-init state, is rejected when replicaSet option is not used.
x		Replicaset server in pre-init state, is not a master nor secondary server, read and write denied.
x		Replicaset pre-init initialization to master server and update master info
x	612-repl-start	Convert pre init replica server to master
x	611-client	Replicaserver rejected when there is no replica option in uri
x		Standalone server rejected when used in mix with replica option defined
x	612-repl-start	Add servers to replicaset
x	613-Client	Replicaset server master in uri, must search for secondaries and add them
x		Replicaset server secondary or arbiter, must get master server and then search for secondary servers