Grid Dynamics

Checklist

For Grid Dynamics blog post — **7 Advanced Practices That Distinguish Effective Mobile Test Automation**

Add 1 point to the score if you answer Yes to a question. Otherwise assign zero points.
How do you read the results? Generally speaking the higher the score, the better the result. It's also good to have no sections with a total zero score. Further explanations of the scoring system are provided in more detail at the end of the checklist.

Question	**Score 1-Yes, 0-No**
Readiness of tests to be run against the production environment	
Environment agnostic - Is it possible to run the same tests against different environments (excluding production) without any changes beyond configurations?	
Production readiness - Do you have a set of functional tests that can be run against the production environment at any time without modifications?	
Avoiding dangerous actions - Are you doing all destructive operations (like database cleaning) in the test framework or setup/teardown phases only and not within the tests' steps?	
Test data decoupling - Is the test data required by the test NOT hardcoded in test steps directly?	
Keep tests green - If you have test runs against the production environment are the results always "green" except cases when real issues are found?	
Regular test runs - If you have a production test subset, do you run it regularly (once per day or more often)?	
Total for readiness of tests to be run against the production environment	
Investment in test data management (TDM)	
Data categorization - Does your TDM solution support data categorization, so you are able to select data types (or business entities, e.g. user, ticket, order, product etc) and their attributes (ticket venue place, ticket venue date etc)?	
Environment unification - Does your TDM solution provide the same interfaces independently of the environment?	
State modification - Is the TDM the only way you use to modify data in a data source (in other words to perform INSERT/UPDATE/DELETE operations over the data source)?	

Data source variety - Does your TDM solution support different data types (DBs, Queues, REST Endpoints and so on) and provide a unified interface to work with them from the tests?	
Mixin mode support - Is there an ability using your TDM solution to add specific data to the data that already exists in a data source?	
Test data generation - Is it possible to generate test data using your TDM solution?	
Total for Investment in test data management	
Proper work with test environments	
Markers in tests - Are you aware of which backend services are used by specific tests or test subsets and are they documented in tests (e.g. by tags)?	
Environment sufficiency - Do you have a single environment where all backend services work correctly (excluding production)? Or, alternatively, do you have a number of environments and it's possible to run different test subsets against different environments to pass the whole regression test suite?	
Environment check-up - Do you run special tests against the backend endpoints to realize whether they are functioning correctly or not?	
Fail-fast paradigm usage - Will your test fail fast if it's clear that some backend services in the test environment are broken?	
Test data preparation - Do you prepare test data on the environments in advance to decrease the time of test runs?	
Versions statistics gathering - Do you gather statistics about components' versions and database schema versions that are deployed on the test environments?	
Total for proper work with test environments	
Using stubs, mocks and fakes	
Mocks usage - Do you use mocks/stubs to emulate work of 3rd party services or non-essential backend services?	
Record-and-play - Do you practice a record-and-play approach for requests/responses between your application and backend services?	
Faked backend approach - Do you create a stateful 'faked backend' to be able to completely get rid of the test environment and use it during local development or in smoke tests?	

Real environment test runs - Do you complement test runs done against mocked/faked backends with test runs done against real backends (e.g. against production)?	
3rd-party services accordance - Is it possible to separate the traffic generated by tests and sent to 3rd party services from the traffic sent from production and generated by real users? (For instance if you would like to collect crash statistics and do not want to mock-up the services).	
Switching in runtime - Is it possible to manage in runtime a configuration of the application from the testing framework, especially which services or environment should be used for the following test subset (faked or not, environment 1 or 2 etc)?	
Total for using stubs, mocks and fakes	
Looking behind the scenes	
Deep-links usage - Do you use a deep-links approach in tests to speed up navigation to the screen you are going to test?	
Configuration updates - Is it possible to update the internal state of an application or its configuration in runtime by calling some 'hidden' methods of the application (e.g. cleanup some form like user profile)?	
Direct access to the backend - Do you use direct calls to the backend to update the data or get supplementary information used by the test?	
Testability improvements - Do you ask developers to add specific methods (backdoors) and screens to improve testability of the application?	
API contract checks - Do you have a set of API level tests that confirm that the API has not changed (especially important if you use mocks/fakes)?	
No backdoors tests - If you have backdoors in the 'debug' version of the application, do you have specific tests to ensure these backdoors are absent in the 'release' version?	
Total for looking behind the scenes	
Continuous optimization of tests and test runs	
Continuous Integration (CI) usage - Do you run your tests on CI regularly?	
Context clean-up - Do you avoid cleaning up the context before/after your test?	
Time measurements - Do you measure the time of every individual test in the suite or at least test substitutes?	
Test suite cleanup. Do you regularly remove or rework old tests, tests with duplicate checks and long-running tests?	

Grid Dynamics

Test statistics gathering - Do you gather different test statistics (including test run time) and generate regular reports that show the trends?	
Proper device management - Are you using cloud device farms and private device farms for device management?	
Total for continuous optimization of tests and test runs	
Coverage measurements	
Functional coverage - Do you have a binding between tests and functional areas they probe?	
Requirements coverage - Do you have a traceability between tests and user stories (if you use scrum, for instance) or requirements?	
Defects coverage - Do you tag tests with defect numbers to provide an ability to easily verify the defects?	
Screens coverage - Do you gather coverage of screens in runtime?	
Transition coverage - Do you gather coverage of transitions between screens?	
Test paths coverage - Do you gather test paths coverage that helps to identify duplicates and non optimal test flows?	
Total for coverage measurements	
Grand total	

0-5 points: *Looks like you are at the very beginning of introducing test automation on the project or your approach requires significant improvements.*
6-16 points: *It could be enough if your mobile application is simple, the number of tests is small and the backend consists of only a few services and it is stable. Otherwise, we recommend to go through the list and implement more things from it.*
17-26 points: *It's quite a good result for mid-sized applications with not very complex backends. However it still could require some improvements, especially if the application develops rapidly and you plan to continue to increase the number of regression tests.*
27-36 points: *It's a very good result and you can feel confident even if you have a complex application under test, complex backends, or unstable environments.*
37-42 points: *You are doing great and we think most of the companies doing mobile test automation would be glad to have your level of experience. Congratulations!*