ORTHOGONAL ARRAYS BASED APPROACH TO USER INTERFACES TESTING

Andronova K., *student, department of Software Engineering, KhNURE*
Kauk V.*, PhD, assistant professor, department of Software Engineering, KhNURE*

User Interfaces testing requires the verification of all page elements and controls and their display on a particular screen size and resolution. These days there is a huge variety of devices used by customers and usually those devices have very different screen properties. It causes the problems when creating test cases since this activity requires a big amount of labor costs for running the scope of such test cases created based on all-pairs approach, that is the most usable combinatorial approach for such verifications. So usual activities such as retest and regression testing take a lot of time to complete them.

The main purpose of the article is to find an approach that could decrease the number of test scenarios to execute in order to test a user interface without decreasing of test coverage level for all possible combinations of devices and screen dimensions.

Application of the combinational approach can help to decrease the number of test cases used in a specific test run. Then it is needed to maximize the coverage by comparatively lesser number of test cases.

Orthogonal array technique is a black box testing technique that is a systematic, statistical way of software testing. It is used when the number of inputs to the system is relatively small, but too large to allow for exhaustive testing of every possible input to the systems. The permutations of factor levels comprising a single treatment are so chosen that their responses are uncorrelated and therefore each treatment gives a unique piece of information. The net effects of organizing the experiment in such treatments is that the same piece of information is gathered in the minimum number of experiments.

So the main point is an identification the correct pair of input parameters. To implement orthogonal arrays testing technique is it necessary to follow the below steps.

1. Identify “Factors” – the independent variables.
2. Identify the “Levels” – values which each variable will take.
3. Search for an orthogonal array that has all the factors from step 1 and all the levels from step 2.
4. Map the factors and levels with the requirement.
5. Translate them into the suitable test cases.
6. Look out for the left over or special test cases.

Considering there are 4 Factors, each of them has 3 Levels. According to all-pairs test approach, the number of test cases needed to ensure the full test coverage is:

$$Test\ cases = 3^4 = 81.$$

So, 81 test cases needed for the full test coverage based on all-pairs technique.

Creating an orthogonal array by following the steps 1 to 6, decrease the number of the test cases needed to get a full coverage (pic. 1).

Experiment no.	Factor A	Factor B	Factor C	Factor D
1	1	1	1	1
2	1	2	2	2
3	1	3	3	3
4	2	1	2	3
5	2	2	3	1
6	2	3	1	2
7	3	1	3	2
8	3	2	1	3
9	3	3	2	1

Picture 1 – Test cases combined using the orthogonal array approach

The example above showed the decreasing of needed test case for 9 times. For orthogonal arrays technique it is true that the more cases exist in all-pair combination set, the more times this number can be decreased due to increasing intersection of pair interactions.

The proposed technique also has limitations for usage for testing purposes that are not related to the user interface since it helps finding only the defects which arise due to the pairs as input parameters. To get the best value from the application of the orthogonal arrays technique it is essential to identify the pairs and the interactions between them accurately.

In this article the problem of the big amount of labor costs and time needed for user interfaces testing on various devices with different screen properties with the help of orthogonal arrays approach. The algorithm of the application of this technique was described, strong and weak sides of the proposed approach were listed as well as limitation of the approach usage. The example describing the decrease of a number of test cases for a specific condition was shown.

References

1. Beizer B. Software Testing Techniques. Delhi: Dreamtech, 2003. 550 p.
2. Kaner C., Folk J., Nguyen H.Q. Testing Computer Software. NY.: Wiley, 1999. 480 p.
3. Black R. Advanced Software Testing – Vol. 1: Guide to the ISTQB Advanced Certification as an Advanced Test Analyst. SB.: Rocky Nook Computing, 2008.486 p.
4. Bath G. Software Test Engineer′s Handbook. SB.: Rocky Nook Computing, 2014. 560 p.
Black R. Advanced Software Testing – Vol. 3: Guide to the ISTQB Advanced Certification as an Advanced Technical Test Analyst. SB.: Rocky Nook Computing, 2015. 480 p.