Software Product Line Test List Generation based on Harmony Search Algorithm with Constraints Support

AbdulRahman A. Alsewari[1], Muhammad N. Kabir[2], Kamal Z. Zamli[3]
IBM Centre of Excellence, Faculty of Computer Systems & Software Engineering
Universiti Malaysia Pahang, Pahang, Malaysia

Khalid S. Alaofi[4]
College of Computer Science and Engineering
Taibah University
Saudi Arabia

***Abstract*—In software product line (SPL), selecting product's features to be tested is an essential issue to enable the manufactories to release new products earlier than others. Practically, it is impossible to test all the products' features (i.e. exhaustive testing). Evidence has shown that several SPL strategies have been proposed to generate the test list for testing purpose. Nevertheless, all the existing strategies failed to produce an optimum test list for all cases. Thus, the current study is aimed to develop a new SPL test list generation strategy based on Harmony Search (HS) algorithm, namely SPL-HS. SPL-HS generates a minimum number of test cases that cover all of the features that are required to be tested based on the required interaction degree (t). The results demonstrate that the performance of SPL-HS is able to compete with the existing SPL strategies for generating test list size.**

Keywords*—Harmony search; computational intelligence; combinatorial testing problem

I. Introduction

A software product line (SPL) is a set of a common software-objects that are collected to handle certain tasks [1]. These software-objects are in accordance with the software features. Testing the interface between all the features is aimed to ensure accurate communication and the data transfer between the software's features. Testing of all the features is a challenge as testing all possible interactions is intractable. Nevertheless, many researchers use the combinatorial testing to generate the test list of SPL products [2].

The main challenge of SPL is to minimize the possible test cases during test case generation with constraints supports [3]. To address this issue, many strategies have been implemented, however, none of these are successful to generate the optimum test list. Johansen et al. adopted the notion of covering arrays in their strategy called SPLCAT [4] in which each column represents one feature and each row represents one product configuration. Furthermore, Microsoft has produced a tool called Pairwise Independent Combinatorial Testing (PICT) [5]. PICT uses random selection to generate a test suite. As an alternate, LOOKUP [6] uses In Parameter Oreder Generation (IPOG) approach combined with Minimum Invalid Tuples (MIT) for testing suite generation. Although these strategies are able to generate test suit, but are not well optimized. Generally, minimizing test suite is an optimization problem. Harmony Search algorithm (HS) has been applied to solve many optimization problems. HS demonstrates an excellent performance in test cases optimization compared to the other optimizations algorithms [7, 9]. Nevertheless, the HS in a previous study [10] failed to demonstrate the support for high system configuration. Therefore, the current study has extended work of a previous study [11] and adopted HS in SPL testing and supported high configurations.

The contributions of this paper are as follows:

- A New Software Product Line Testing Strategy has been developed based on HS, called (SPL-HS).
- The constraint combinations of the features have been addressed by carrying out the test cases .

The rest of the paper as: Section 2 will illustrate the SPL background, Section 3 will explain the proposed strategy, Results and discussion will be presented in Section 4, in the last section, the conclusion will be presented.

II. SPL Background

For testing a SPL, there is a need for testing all possible interaction between features. Fig. 1 illustrates the example of Smartphone's Features. Most of the Smartphones like Samsung, IPad, and Nexus 7 are under a similar product line because the devices share some common features such as Wi-Fi, Sim card, Bluetooth, GP, and etc. As such, for testing the interaction between such smartphone, each feature represents as ON or OFF, where ON indicates that the feature is presented in the new product while OFF indicates the opposite. Table 1 demonstrates the values for only three features (i.e. Wi-Fi, Bluetooth, and GPS). There are 8 test cases were applied for testing this feature as shown in Table 2 (i.e. Exhaustive testing). For four features, there are 16 test cases are required to test all the combinations. Hence, generating test cases is NP-hard problem. Normally, a SPL contains more features. For testing the combinations for 20 features, then the generated test cases are 1048576 test cases. If each test case requires five minutes, testing 20 features will take 5,242,880 minutes (around 87381 hours) for exhaustive testing.

Combinatorial Testing (CT) is a method for generating covering an array (CA) test suite with the consideration of interactions between features of SPL [12]. On that account, during testing any software that has several inputs of features, it is not possible to trigger errors or bugs with any combination of the system features. Therefore, the testing

requires interaction strength that can reduce the number of test cases based on the identified requirements or based on tester experience.

Fig. 1. Features of Smartphone.

TABLE I. FEATURES SOFTWARE PRODUCT LINE EXAMPLE

Feature	**Wi-Fi**	**Bluetooth**	**GPS**
Value	On	On	On
	Off	Off	Off

TABLE II. EXHAUSTIVE TESTING TEST LIST

No	**Wi-Fi**	**Bluetooth**	**GPS**
1	*On*	*On*	*On*
2	*On*	*On*	*On*
3	*On*	*On*	*On*
4	*On*	*On*	*On*
5	*On*	*On*	*Off*
6	*On*	*On*	*Off*
7	*On*	*On*	*Off*
8	*On*	*On*	*Off*

Each feature of the smartphone is treated as an input parameter with value on and off as shown in Table 1. The exhaustive test list consists of 2 x 2 x 2, which are 8 test cases as shown in Table 2.

The process of SPL-HS in 2-way interaction strength (i.e. $t = 2$) is described as below:

First, the interactions between the features are: Wi-Fi x Bluetooth (2 x 2 = 4 combinations), Wi-Fi x GPS (2 x 2 = 4 combinations), and Bluetooth x GPS (2 x 2 = 4 combinations).

Then, SPL-HS is able to generate a test list with 4 test cases or more but less than 8 test cases.

III. PROPOSED STRATEGY SPL-HS

This paper proposes a new t-way strategy to generate test cases for SPL testing based on HS with constraint support called (SPL-HS). On that account, HS uses to select only valid products from all possible products. The following steps illustrate on how HS applies in SPL testing.

The implantation of the proposed strategy involves three main parts: a) interaction list generation, b) constraint handling and c) test case generation.

A. Interaction List Generation

In this stage, the SPL-HS will generate all possible interaction between the features according to the interaction degree (t) as in Table 3.

Each digit of a binary number represents a single possible interaction. The binary number 11100, represents the interaction combination index for WiFi, Camera, and GPS, while 11010 represents the interaction combination index for WiFi, Camera, and Media and etc. (see Table 3).

In SPL, each feature has two possible values, namely On or Off (i.e. selected or not selected in the new product). Table 4 demonstrates the example of the interaction elements list of the first index 11100 (i.e. WiFi, GPS and Camera). Moreover, there are additional interaction element lists available for the other indexes (11010, 11001, 10110, 10101, 10011, 01110, 01101, 01011, 00111).

TABLE III. INTERACTION LIST OF A SMARTPHONE PARAMETERS

WiFi	Camera	GPS	Media	Message	Interaction	
X	x	x			WiFi, Camera, GPS	11100
X	x		x		WiFi, Camera, Media	11010
X	x			x	WiFi, Camera, Message	11001
x		*x*	*x*		WiFi, GPS, Media	10110
x		*x*		*x*	WiFi, GPS, Message	10101
x			*x*	*x*	WiFi, Media, Message	10011
	x	*x*	*x*		Camera, GPS, Media	01110
	x	*x*		*x*	Camera, GPS, Message	01101
	x		*x*	*x*	Camera, Media, Message	01011
		x	*x*	*x*	GPS, Media, Message	00111

TABLE IV. INTERACTION ELEMENTS LIST FOR COMBINATION OF WIFI, GPS AND CAMERA (11100)

No.	WiFi	GPS	Camera
1	*On*	*On*	*On*
2	*On*	*Off*	*Off*
3	*On*	*On*	*Off*
4	*On*	*Off*	*On*
5	*Off*	*On*	*Off*
6	*Off*	*Off*	*On*
7	*Off*	*On*	*On*
8	*Off*	*Off*	*Off*

B. Constraints Handling

There are two types of constraints in SPL testing; required and excluded constraints. Constraints in SPL fix certain combinations of features in final test suite whether these constraints are excluded or required.

The required constraints are combinations of features that needed for the final test suite. Specific combinations are carried out to test the smartphone product, for example, WiFi feature must be tested along with GPS. Therefore, at least one test case that contains WiFi and GPS with the values (On, On) is required to be included during test suite generation.

Excluded constraints are combinations of features that are required to be excluded from the final test suite.

For example, in another testing, to test the smartphone product, Media features could not be operated by Camera features; therefore the combination of Media and Camera is excluded from the final test suite (Fig. 2).

At this stage, strategy lists of required combinations and a list of excluded combinations have been proposed. Then, each test case that has been generated was checked whether it contains the required combination to be added to the final test suite. In addition, the test case was checked if it contains unwanted combination to be excluded from the test case. For example, when the parameter value equal to 4, interaction degree in = 2, and the value of the excluded constraint is (x01x), "x" represent no constraint value in this feature and the combination that involves second parameter and third parameter with values of 01 should be deleted from the test cases.

C. Test Case Generation

Based on the concept of HS, the test suite generation steps in SPL-HS are listed as below (see Fig. 3):

- Initialization of HS's parameters such as the harmony memory size (HMS), the harmony memory consideration rate (HMCR), the pitch adjustment rate (PAR) and the iteration.
- Construction of the harmony memory (HM) with random test cases considering the constraint combinations based on HMS.

$$T=x_1,x_2,x_3,\ldots.x_n \quad (1)$$

$$x_i = Random * (UB-LB) \quad (2)$$

where T represents the test case, x_i represents the value of the feature $I_.$.

- Improvement of the test list by either randomly generate test case or adjusting the selected existing test case from the HM with consideration the constraint combinations.
- Updating HM by replacing the worst test case in HM with the new test case generated from the improvement step iii.
- Repetition of steps iii, and iv until meeting the exit criteria of the improvement.
- Add the best test case in the HM to the final test list.
- Repetition of steps ii to vi until all the interactions in the interaction lists are covered.

IV. RESULT AND DISCUSSION

The performance of the SPL-HS were evaluated by conducting the following experiments: Firstly, the test cases were generated for SPL with constraints supports. Secondly, the test cases were generated for several system configurations. In both experiments, the results of SPL-HS's were compared with the results of existing strategies.

The SPL-HS run in the Java platform on an Asus A45 laptop with the specification of Intel Core i7-2450M CPU 6GB DDR3, SATA 500GB Hardisk and run on operating system Windows 10. Each experiment was repeated for 30 times and carried out to obtain the average and the minimum results for SPL-HS.

The SPL-HS parameters were initialized based on a previous study [9] as follows: HMS size was 100 test cases, HMCR with 0.7, iteration of improvisation was 1000, PAR was 0.5.

The future work should investigate on supporting higher than 2-valued parameter, which would allow the strategy to be applied on other combinatorial testing problems. Moreover, input-output feature, which allows the tester to define the combinations for generating the test case should also be evaluated in the future.

A. Experimental Result on SPL with Constraints Supports

In this section, a selected case study from SPLOT [13] was used. The study features repository for the feature model of the video player. The case study contains 71 features (i.e. 23 are mandatory features and 12 are optional features). In this model, certain features were included. Therefore excluded constraints were defined prior to the generation of the test suite. The features involved are (F5, F6, F7), (F9, F10 …… F14), (F22, … F29), (F32 … F42), (F44 …… F50) could not be *OFF* simultaneously because at least one of them in each set must be *On*.

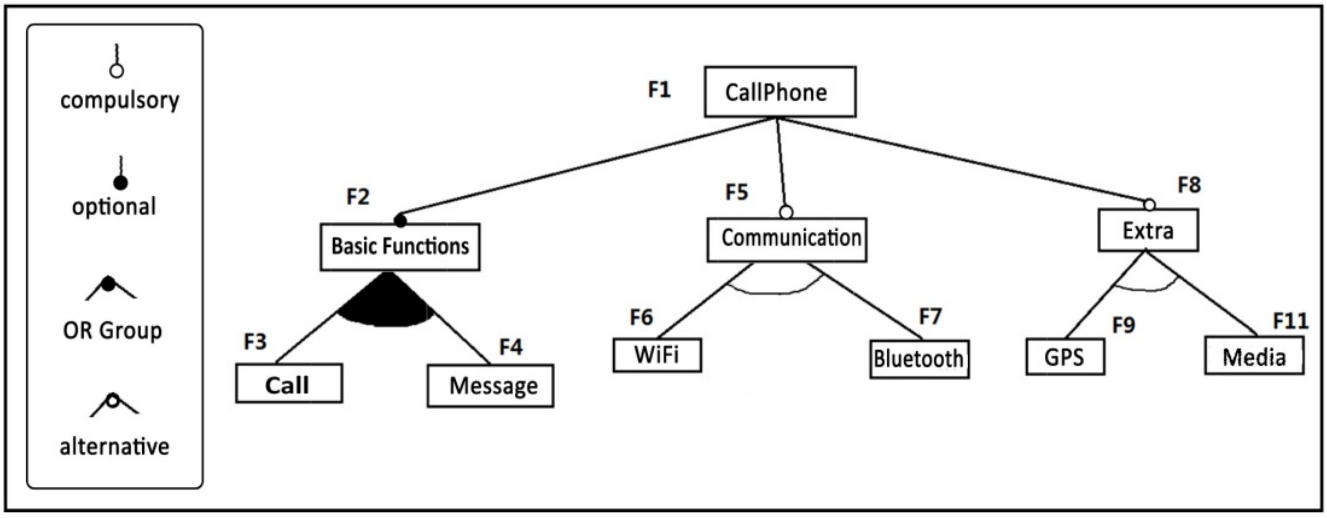

Fig. 2. Feature Model of Smart Phone Example.

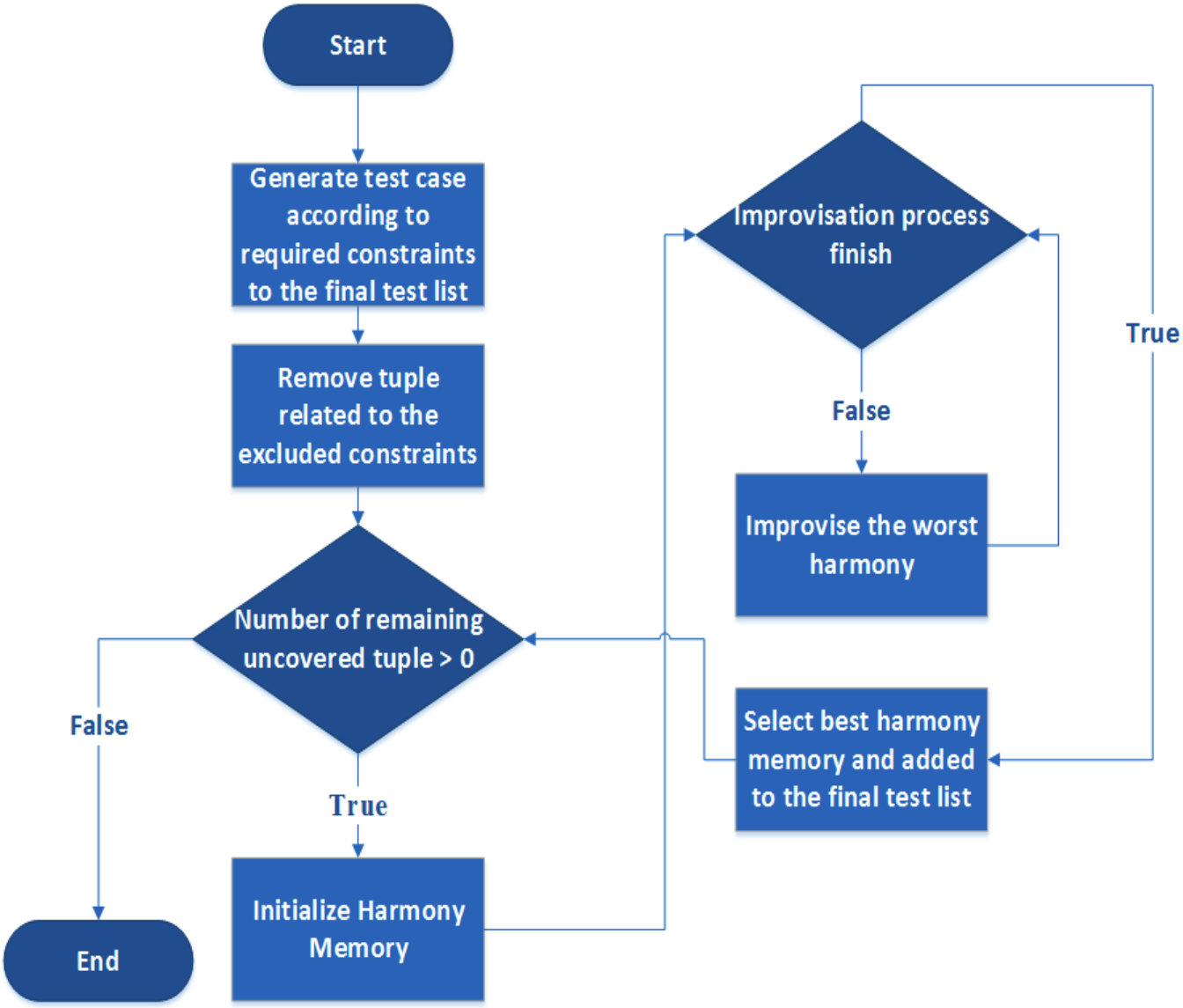

Fig. 3. Implementation of Harmony Search Algorith.

TABLE V. RESULT OF COMPARING SPL-HS WITH EXISTING STRATEGIES WITH CONSTRAINTS SUPPORTS

Combination Degree (t)	PICT	SPLC	LOOKUP	SPLBA	SPL-HS
2	15	16	18	**13**	**13**
3	47	47	**39**	49	46
4	N/A	N/A	N/A	N/A	**153**

Table 5 demonstrates that the proposed strategy is able to produce a minimum test suite size in all cases compared with other strategies. In this case, SPLBA and SPL-HS produced the best result (i.e. 13 test cases), when t = 2. LOOKUP produced the best test size (i.e. 39 test cases) when t = 3. SPL-HS generated superior result compared to SPLBA, SPLC and PICT, which is 46 test cases. For t = 4, SPL-HS managed to produce the result of 153 test cases, however, the results were unavailable from the other strategies. In general, SPL-HS produced a superior results compared to other strategies with supporting for t = 4.

B. Experimental Result on T-way

The proposed strategy was compared with the existing t-way strategy to evaluate the performance of SPL-HS strategy during the t-way testing. The results were obtained from a previous study that a test generation research tool called LOOKUP performs better than the existing test generation tools in term of test size and execution time [6].

Table 6 demonstrates that SPL-HS has produced superior results in most of the cases. Nevertheless, there was no significant difference between SPL-HS and IPOG-F in other cases, which IFOG-F has produced reliable results. Based on the balancing between the local search and the global search in HS, SPL-HS has demonstrated an ability to generate superior or at least same result as IPOG-F for lower interaction degree (t). Table 6 demonstrates that SPL-HS has achieved 26 out of 30, while IPOG-F achieved 17 out of 30. Hence, SPL-HS has worked efficiently with a higher interaction degree while IPOG-F produced poor results compared to SPL-HS. This is mainly due to SPL-HS search for best test list in local search and global search.

V. CONCLUSION

The current study proposed a new strategy for SPL testing, known as SPL-HS. SPL-HS adopted Harmony Search as the optimization algorithm and generated test cases for SPL that supports constraints for both required constraints and excluded constraints.

SPL-HS is the first strategy that adopted HS as the core implementation for generating a test suite for SPL that is capable to support t equal 4.. The SPL-HS has superior performance in comparison with existing SPL strategies such as PICT, SPLC, LOOKUP and SPLBA. SPL-HS produced superior result compared to IPOG-F results when t is equal to 4, while it failed to produce satisfactory results when t is equal to 2 and 3.

TABLE VI. RESULT COMPARING SPL-HS WITH IPOG-F

T	Parameters	IPOG-F	SPL-HS (best)	SPL-HS (Avg)
t= 2	10	**8**	**8**	9
	20	**10**	**10**	10.6
	30	**11**	**11**	11.3
	40	**11**	**11**	12.2
	50	**11**	**11**	12.9
	60	**12**	**12**	13.2
	70	**12**	**12**	13.6
	80	**13**	**13**	14.1
	100	**13**	**13**	15.1
	200	**15**	**15**	17.8
	300	**16**	**16**	20
t= 3	4	9	**8**	8.5
	8	17	**15**	15.99
	12	19	**18**	19.3
	16	**22**	**22**	22.59
	20	**25**	**25**	25.5
	24	**26**	28	28.5
	28	**28**	30	30.6
	32	**31**	32	32.4
t= 4	5	22	**16**	19
	6	**26**	27	27.79
	7	32	**28**	30.5
	8	34	**32**	33.09
	9	37	**33**	36.69
	10	41	**36**	39.3
	11	43	**42**	46,1
	12	47	**40**	43.4
	13	49	**44**	46.19
	14	52	**46**	48.2
	15	53	**49**	50.4

ACKNOWLEDGMENT

This research is funded by RDU180367, "Enhance Kidney Algorithm for IoT Combinatorial Problem", and (FRGS/1/2018/ICT05/UMP/02/1, RDU190102), "A Novel Hybrid Kidney-inspired algorithm for Global Optimization".

REFERENCES

[1] T. Thüm, S. Apel, C. Kästner, I. Schaefer, and G. Saake, "A Classification And Survey of Analysis Strategies For Software Product Lines," ACM Computing Surveys (CSUR), vol. 47, p. 6, 2014.

[2] P. Clements and L. Northrop, "A Framework For Software Product Line Practice," SEI interactive, vol. 2, 1999.

[3] I. do Carmo Machado, J. D. McGregor, and E. Santana de Almeida, "Strategies For Testing Products In Software Product Lines," ACM SIGSOFT Software Engineering Notes, vol. 37, pp. 1-8, 2012.

[4] M. F. Johansen, Ø. Haugen, and F. Fleurey, "Properties Of Realistic Feature Models Make Combinatorial Testing Of Product Lines Feasible," in Model Driven Engineering Languages and Systems, ed: Springer, pp. 638-652, 2011.

[5] J. Czerwonka, "Pairwise testing in the real world: practical extensions to test-case scenarios," in Proceedings of 24th Pacific Northwest Software Quality Conference, Citeseer, 2006, pp. 419-430.

[6] L. Yu, F. Duan, Y. Lei, R. N. Kacker, and D. R. Kuhn, "Combinatorial Test Generation For Software Product Lines Using Minimum Invalid Tuples," in High-Assurance Systems Engineering (HASE), 2014 IEEE 15th International Symposium on, pp. 65-72, 2014.

[7] K. Z. Zamli, A. R. Alsewari, and B. Al-Kazemi, "Comparative Benchmarking of Constraints T-Way Test Generation Strategy Based on Late Acceptance Hill Climbing Algorithm," International Journal Software Engineering Computer Science(IJSECS), vol. 1, pp. 14-26, 2015.

[8] K. Z. Zamli, F. Din, G. Kendall, and B. S. Ahmed, "An Experimental Study of Hyper-heuristic Selection and Acceptance Mechanism for Combinatorial T-Way Test Suite Generation," Information Sciences, vol. 399, pp. 121-153, 2017.

[9] K. Z. Zamli, A. R. Alsewari, and M. H. M. Hassin. "On Test Case Generation Satisfying the MC/DC Criterion," International Journal of Advances in Soft Computing & Its Applications, vol 5, pp. 104-115, 2013

[10] A. R. A. Alsewari and K. Z. Zamli, "A Harmony Search Based Pairwise Sampling Strategy for Combinatorial Testing," International Journal of Physical Sciences, vol. 7, pp. 1062-1072, 2012.

[11] A. R. A. Alsewari and K. Z. Zamli, "Design and Implementation of a Harmony-Search-Based Variable-Strength T-Way Testing Strategy with Constraints Support," Information and Software Technology, vol. 54, pp. 553-568, 2012.

[12] C. Nie and H. Leung, "A Survey of Combinatorial Testing," ACM Computing Surveys (CSUR), vol. 43, p. 11, 2011.

[13] M. Mendonca, M. Branco, and D. Cowan, "SPLOT: Software Product Lines Online Tools," in Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, pp. 761-762, 2009.