SONATA

SONATA SOFTWARE

Software Testing Optimization using Combinatorial/Pairwise technique

Author – madhusudan kl
Co-Authors – mahesh c & raj k

Objective

SONATA SOFTWARE

In today's dynamic market, Business solutions landscape is heterogeneous!

✓ To integrate different products with large scale data variants requires endless testing.

✓ Need of the hour – “Save time, effort and find bugs effectively”.

Is exhaustive testing a realistic method to meet the customer expectations?
How to achieve greater coverage in fewer tests?

✓ Solution to this challenge is Combinatorial/Pairwise test design.

✓ Combinatorial test design technique helps tester to identify the high probable bug prone candidates, which means a small set of test cases.

Benefit for customer = Fewer tests with greater coverage!

Topics

SONATA
SONATA SOFTWARE

1. Combinatorial Technique - Introduction
2. How to Use/implement Technique?
3. Applicability of Combinatorial Technique
4. Case Studies
5. Challenges of Combinatorial Technique
6. Executive Summary

Introduction - Combinatorial Technique

SONATA SOFTWARE

Test case selection poses an interesting dilemma for the software professional.

Manual testing can only show the existence of defects and never their absence, and that exhaustive testing quickly becomes impossible.

However, testing is necessary. Being intelligent about which test cases you choose can make all the difference between

✓ Endlessly executing tests that just aren't likely to find bugs and don't increase your confidence in the system and

✓ Executing a concise, well-defined set of tests that are likely to uncover most (not all) of the bugs and that give you a great deal more comfort in the quality of your software.

Combinatorial or pairwise testing is a systematic, statistical way of testing pair-wise interactions. It provides representative (uniformly distributed) coverage of all variable pair combinations. It helps in testing combinations of configurable options.

Combinatorial or pairwise is a method of software testing that, for *each pair* of input parameters to a system (typically, a software algorithm), tests all possible discrete combinations of those parameters. Using carefully chosen test vectors, this can be done much faster than an exhaustive search of all combinations of all parameters, by "parallelizing" the tests of parameter pairs

Combinatorial Technique

SONATA SOFTWARE

Pairwise (2way) testing

All-pairs testing or Pairwise testing is a combinatorial software testing method

✓ For each pair of input parameters to a system (typically, a software algorithm), tests all possible discrete combinations of those parameters.

✓ Using carefully chosen test vectors, this can be done much faster than an exhaustive search of all combinations of all parameters, by "parallelizing" the tests of parameter pairs.

✓ The number of tests is typically O (nm), where n and m are the number of possibilities for each of the two parameters with the most choices.

✓ The reasoning behind all-pairs testing is this: the simplest bugs in a program are generally triggered by a single input parameter.

✓ The next simplest category of bugs consists of those dependent on interactions between pairs of parameters, which can be caught with all-pairs testing.

Continued in 6th slide…….

Combinatorial Technique

SONATA SOFTWARE

Pairwise (2way) testing …..continued from 5th slide

✓ Bugs involving interactions between three or more parameters are progressively less common, whilst at the same time being progressively more expensive to find by exhaustive testing.

✓ Pairwise testing is a reasonable cost-benefit compromise between - often computationally infeasible higher-order combinatorial testing methods, and less exhaustive methods which fail to exercise all possible pairs of parameters

✓ Because no testing technique can find all bugs, all-pairs testing is typically used together with other quality assurance techniques such as unit testing, symbolic execution, fuzz testing, and code review

Combinatorial Technique

SONATA SOFTWARE

T-Way (n ways)

T- Way is an extension of Pairwise Test Case generator method

✓ Provides increased coverage of combinations of values from the provided Test Parameters

✓ In situations where 3-way/4-way/5-way combinations can reveal defects, this tool will be rightly helpful to generate tests that cover them

✓ By increasing the coverage strength to 3-way and 4-way between critical test parameters, several testers were successful in identifying hard-to-find defect/s.

How to use/implement technique?

SONATA SOFTWARE

Identify user actions / Inputs

During the requirement analysis all possible user actions and / or input data must be identified for each requirement.

✓ Output from this phase will be user actions and/or inputs.

✓ For every user action identify the expected behavior.

✓ Group and / or ungroup user actions/inputs to form input for combinatorial test generation.

✓ Use tool/s (MSPICT/All Pairs.exe/AETG Web services) to generate the test combinations for Pairwise and T-Way.

✓ Review the test combinations and remove invalids. (Will not be more than 0.5%).

✓ Start test case design in the following order.

a. Pairwise test combinations.
b. T-Way test combinations.

Continued in 9th slide……..

How to use/implement technique?

Continued from 8th slide..

✓ Design 1st test case as Template by using parameterization technique. I.e. make sure all user actions/ inputs are parameterized.

✓ Example: departure date, duration, departure airport …etc. For each step expected results should be clearly defined.

✓ Just need to change the parameter values and expected results for remaining all test cases.

✓ Review the test cases to ensure all actions parameterized and all parameter values changed according to the Pairwise / T-Way test combinations.

✓ Execute the test cases.

Applicability of Combinatorial Technique

SONATA SOFTWARE

Following are the scenarios where combinatorial technique can be applied.

To test complex domain size

Testing of complex domain size problem can be easily illustrated.

✓ A GUI (Graphical User Interface) based application has many operations that need to be tested.

✓ A very small program such as Microsoft WordPad has 325 possible GUI operations/combinations.

✓ In a large program, the number of operations can easily be an order of larger magnitude

Applicability of Combinatorial Technique

Following are the scenarios where combinatorial technique can be applied.

To deal with sequences of pairing combination

Some functionality of the system may only be accomplishable by following some complex sequence of GUI events.

✓ For example, to open a file a user may have to click on the File Menu and then select the Open operation, and then use a dialog box to specify the file name, and then focus the application on the newly opened window.

✓ Obviously, increasing the number of possible operations increases the sequencing problem exponentially.

✓ This can become a serious issue when the tester is creating test cases manually

Applicability of Combinatorial Technique

Following are the scenarios where combinatorial technique can be applied.

To deal with regression testing scenarios

✓ Regression testing becomes a problem with GUIs as well.

✓ This is because the GUI may change significantly across versions of the application, even though the underlying application may not.

✓ A test designed to follow a certain path through the GUI may not be able to follow that path since a button, menu item, or dialog may have changed location or appearance

✓To deal with all single mode, double mode and multimode faults

Case Study - One

Case Study - One

A complex Retail application with 6 fields x 16 possible values likely generates 324 manual test cases, is selected for combinatorial testing to increase confidence in test design/coverage.

#	Field Name	Field Type	# Values	Comments
1	Country	Drop down list	3	UK, US, Canada
2	Search Type	Option button	2	Language, Generic
3	Language	Drop down list	2	English, French
4	Title	Text Box	3	Valid, Invalid, Special Characters
5	Distribution Part #	Text Box	3	Valid, Invalid, Special Characters
6	Incentive	Drop down list	3	All, Promo, Non-Promo

Test data of Retail application page

Case Study - One

As seen in Case Study – One test data, the test data of Retail application page consists of 6 field's, 3 different field type's and field names supports 16 different static values.

#	Description	Values	Manual TC*	Allpairs TC*	Effort Savings	AETG/ MS PICT TC*	Effort Savings
1	Retail Page	3x2x2x3x3x3	324 no's	36 no's	88.89%	12 no's	96.29%

Test case effort comparison table for Retail Application Page

Note: Please refer to formula calculation as mentioned in Notes section

Case Study - One

Summary of Case Study - One

✓ 324 test cases are generated manually for 96 possible manual combinations

✓ Allpairs.exe freeware tool generates 36 pair wise test cases

✓ Allpairs.exe freeware tool results in 88% effort savings in comparison to manual test cases

✓ AETG web services/MS PICT tool generates 12 pair wise test cases

✓ AETG web services/MS PICT tool results in 96.3% effort savings in comparison to manual test cases

Case Studies

Results of Case Study - One

✓ As evident, test case generation optimality is achieved in Case studies of Retail application page.

✓ The effort savings ranges from 88 to 100% depending upon the complexity of the scenario involved.

✓ As seen in first Case Study, combinatorial tool aided software test case generation is mathematically proven and is used worldwide and industry experts are of the opinion that combinatorial tool aided test case generation is accurate/optimal/precise.

Case Study - Two

Case Study - Two

A reputed University home page is used by its students/faculty/learning centre's in the real time for education purposes.

✓ Its Home page with a combination of 9 fields and with 30 possible values generates 5184 possible manual test cases.

✓ Since this page forms an important feature of the application and with reasons to increase the test coverage and to instill the confidence in application.

✓ This page is selected for combinatorial case study.

Case Studies

#	Field Name	Field Type	# Values	Comments
1	Name	Text Box	3	Valid, Invalid, Blank
2	Email ID	Text Box	4	Valid, Invalid, Special, Blank
3	Mobile Number	Text Box	3	Valid, Invalid, Blank
4	Last Degree	Drop down list	6	Last Degree, 12, UG, PG Diploma, Advanced Diploma
5	Select Disciple	Drop down list	8	Select Discipline, Allied Health, BT, Hospitality, IT, Journalism, Management, Telecom
6	Interested Course	Drop down list	3	MBA, MCA, BCA
7	Select State	Drop down list	1	State1
8	Select City	Drop down list	1	City1
9	Select Location	Drop down list	1	Location1

Test Data for University Home page

Continued in 19th slide

Case Studies

As seen in University Home Page Test Data table , home page consists of 9 field's, 2 different field type's and the field names supports 30 different dynamic values.

#	Description	Values	Manual TC*	Allpairs TC*	Effort Savings	AETG/ MS PICT TC*	Effort Savings
1	University Home Page	3X4X3X6X8 X3X1X1X1	5184 no's	113 no's	97.0 %	101 no's	98.06 %

Test Case Effort comparison table for University Home Page

Note: Please refer notes section as mentioned below

Case Studies

SONATA SOFTWARE

Case Study - Two

✓ 5184 possible manual test cases are generated

✓ Allpairs.exe freeware tool generates 113 pair wise test cases

✓ Allpairs.exe freeware tool results in 97% effort savings in comparison to manual test cases

✓ AETG web services/MS PICT tool generates 101 pair wise test cases

✓ AETG web services/MS PICT tool results in 98.06% effort savings in comparison to manual test cases

Case Studies

Results of Case Study - Two

✓ As evident, test case generation optimality is achieved in Case studies of Retail application page, and in University Home page.

✓ The effort savings ranges from 88 to 100% depending upon the complexity of the scenario involved.

✓ As seen in the Case Studies, combinatorial tool aided software test case generation is mathematically proven

✓ Is used worldwide and industry experts are of the opinion that combinatorial tool aided test case generation is accurate/optimal/precise.

Case Studies

SONATA SOFTWARE

Case Study - Three

We assessed the quality of the combinatorial methodology in a project. We started scripting using regular approach to design the test cases and in parallel designed Combinatorial test scenarios.

Plan is to test code drop1 with the test cases developed using regular approach and in parallel execute the combinatorial test scenarios to find out the difference of quality.

Code drop1 we have executed all regular test cases and performed Adhoc testing which resulted in 77 defects

Case Studies

Defect Priority	
Urgent	13
High	21
Medium	27
Low	15
Total	77

Defect Priority	
Urgent	22
High	26
Medium	35
Low	25
Total	108

108 defects reported using Combinatorial testing which are 31 additional defects

Results of Case Study Results

✓ With manual/risk based test approach we couldn't find all the defects since it is not possible to cover all the combinations in manual test cases.

✓Combinatorial test scenarios produced all urgent and high defects in 1st and 2nd day test execution.

✓No surprises after go-live.

Challenges of Combinatorial Testing

SONATA SOFTWARE

Combinatorial test case generation technique can be extensively used for generating test cases for Unit Testing, Smoke Testing, Usability Testing, Browser Compatibility testing, Functional Testing, End-End testing, Performance Testing and Security testing

Following challenges were faced while implementing combinatorial technique:

✓ **Determining test data** – For combinatorial technique to be successful, all +/- ve data should be initially determined and it should be fed to the tool

Overcoming this Challenge – Take a screenshot of the AUT and list out all the possible +/- ve test data for the available fields in the applications page (under test)

✓ **Ensuring coverage does miss any data fields** – For combinatorial technique to be successful, it is very important to make sure all the relevant data fields and its values are covered in test data

Overcoming this Challenge – Make sure to include all relevant data fields in AUT for combinatorial coverage.

Summary

SONATA SOFTWARE

✓ In the real time software development life cycle, software testing phase consumes nearly 30-40% man hours of the entire development efforts.

✓ With tight delivery schedules and stiff market competition, alternative improvement techniques for reducing the time consumption in software testing phase is the need of the hour.

✓ Due to complexities in the domain, manual test case coverage for the voluminous applications would be cumbersome and time consuming; there is likely risk and chance of a functionality/defect being leaked to the end customer.

✓ Reduction in test cycle time has been successfully established if combinatorial technique is used for test case generation.

✓ Savings in efforts ranging from 88-100% is the outcome.

✓ Also we can note that how a proven mathematical technique called Combinatorial Technique, can be utilized in optimizing test case generation and by improving the confidence level of the correct application working, before it's released to the end customer/s.

SONATA

SONATA SOFTWARE

Thank You