直交表を利用したソフトウェアテスト
－ＨＡＹＳＴ法－

山本 訓稔† 秋山 浩一‡

†‡富士ゼロックス株式会社 商品評価部 システム検証センター

〒213-8508 神奈川県川崎市高津区坂戸 3-2-1 KSP 9A5

E-mail: †Kunitoshi.Yamamoto@fujixerox.co.jp, ‡Kouichi.Akiyama@fujixerox.co.jp

あらまし 直交表を利用した新しいソフトウェアテスト手法(HAYST 法)を確立した。この手法を用いることで、従来と比較し、4.5 倍のバグが検出でき、また、テスト設計にかける工数を減らすこともできた。直交表をソフトウェアテストに用いる目的はそのユニークな性質を利用することにある。直交表は、２つの因子（パラメータ）間の総組合せを保証する。多くのバグが２因子間の組合せにより発生していることから、直交表の利用はソフトウェアの品質保証にとても有効であることが分かった。

キーワード 直交表、機能組合せテスト、全ペア、品質工学（田口メソッド）

Application of Orthogonal Array to Software Testing.
－HAYST Method－

Kunitoshi YAMAMOTO† and ‡Kouichi AKIYAMA‡

†‡Fuji Xerox Co., Ltd. 3-2-1, Sakado, Takatsu-ku, Kawasaki-shi, Kanagawa, 213-8508 Japan

E-mail: †Kunitoshi.Yamamoto@fujixerox.co.jp, ‡Kouichi.Akiyama@fujixerox.co.jp

Abstract We established a new SW testing method (HAYST) using Orthogonal Array (OA). This method can assure the SW quality 4.5 times as much as that of past method. And it can reduce the cost of test design. The purpose of applying OA to SW testing is to use its unique feature. OA can assure combinations between two factors in it. This is useful for SW quality assurance, because most of SW bugs are caused by pair-wise combinations of its factors.

Keyword Orthogonal Array, combination test, pair-wise, Taguchi's method.

1. はじめに

1.1. 直交表のソフトウェアテストへの利用状況

ブラックボックステストを実施する場合、機能の組合せを確認する必要がある。しかし、機能の組合せ確認を「無条件に全ての機能を組合せる」という方針で実施すると、テスト項目数は指数関数的に増加しテストを実施することが不可能となる。この組合せの発散を抑える方法の一つとして「直交表を活用したテスト設計」を行うことが有効であると言われている。

直交表は、実験計画法とともに 1940 年代より日本でも使われはじめ、1957 年に田口玄一氏により「実験計画法」[1]が出版されたことを契機とし、広く使われるようになった。

ソフトウェアへの適用においても、ブラックボックステストの一手法として「要因分析法」や「実験計画法」という名前で紹介されている[2][3][4][5]。

また、1999 年に田口玄一氏からも「ソフトウェア評価に対する品質工学の適用」という観点で提案が出され[6]、その試行報告もある[7]。

一方海外では、Cohenらによって、直交表実験と数学的にほぼ等価な全ペアを確認することとカバレッジの関連について研究がされ「Block Coverage(C0)が 92%、Decision Coverage(C1)が 85%と同等の網羅性」と計測されている[8]。これは、ブラックボックステストとしては驚きの値であり、全ペアテストの重要性が認識された。

これら直交表適用の狙いは、全ての機能組合せをテストするという方針から「少ないテスト回数で２機能間の全組合せを作ること、また、できるだけ３機能間の組合せを多く作ることという方針」に切り替えることにある。例えば、ON と OFF の２種類の値を持つ機能が 15 個ある場合、直交表に割り付けると、テスト項目数は 16 となる。この時、２機能間の組合せは 100%

現れ、3機能間の組合せも50%出現する。これをもし、直交表を用いずに全ての組合せをテストしようとすると2の15乗、すなわちテスト項目は32,768項目（直交表を使用した場合の2,048倍）となる。

直交表テストの欠点は3機能以上の多次元の組合せ網羅性に対しての保証が無い点と、多次元の組合せを保証しようとするとテスト設計が難しくテスト項目数も増えテストの実現性がなくなる点にある。

ここであらためて良いテストケースとは何かについて考えてみる。Myersは、著書[9]の中で、「良いテスト・ケースとは、まだ発見されていないエラーを検出する確率の高いものである」と述べている。言い換えると「より危険なバグから優先的に同じ工数でたくさん見つけることができるもの」が良いテスト・ケースであると言える。一般的に2機能間で発生するバグの方が多次元の組合せで発生するバグよりも発生頻度が格段に高いことからより危険なバグであると言ってよい。当社で過去の不具合を分析した結果においても、重大な不具合のほとんどが2機能間の組合せで発生していると分かった。そこで、筆者らは上記多次元の組合せにおける直交表適用の欠点についてはそれほど問題にならず、むしろ適切なテスト優先度になるのではないかと考え、SWテストへの直交表の適用を検討してきた。

1.2. 直交表適用における問題点

当社のプリンタドライバのソフトウェア機能組合せ評価に直交表を製品評価に適用したところ、そのままでは全体の1割程度しか割り付けられないことがわかった。その原因は各パラメータ（因子）の選択肢（水準）が非常に多いことと、ソフトウェア特有の禁則の複雑さにあった。

従来の直交表実験はL36直交表を用いることが推奨されている[6]。L36直交表は、2水準11列、3水準12列の混合型直交表であるが、多水準を割り付けることができないためプリンタドライバのような多因子・多水準をもつソフトウェアについて、そのまま割り付けることができなかった。また、代表水準を選択し、L36直交表に割り付けたとしても、9割程度のテスト項目に禁則条件があらわれ、テストが進まなかった。

筆者らは十分なバグ出しが終わったソフトウェアに対して、最終品質確認という意味でL36直交表を用いてテストすると、工数対効果は非常に高くなるが、バグを多く含んだソフトウェアに適用した場合、バグを見逃してしまう危険があると考えている。

本研究では、上記問題点に対して取った施策と、実際のソフトウェア評価へ適用した結果および今後の取り組みについて報告する。

2. HAYST法

筆者らは、1で述べた問題点を解消し、直交表のもつ利点を生かした新しいテスト手法を確立しHAYST法（ヘイストホウ）と名付けた。HAYST法の持つ特徴は次の4点である。

① 多因子・多水準に対応するために変形しやすい2水準系の直交表を利用
② 禁則回避処理を考案しツールに実装
③ 組合せ網羅率をテスト品質指標として採用
④ バグの即時原因解析

次節より、上記特徴の詳細について述べる。

2.1. 多因子・多水準への対応

筆者らが取り扱うソフトウェアは、例えば「用紙サイズ選択」という一つの機能に対して、A4, A3, B4, B5, B6, Letter, 封筒, 葉書...と多値を持つものが多数存在している（図1）。

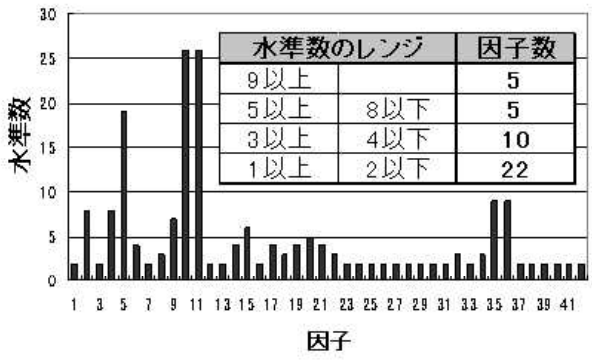

図1　因子別水準数の状況

これまでの直交表テストの方法は、これらについて、ドメイン分割（A系用紙、B系用紙、欧系用紙等）し、同値クラスの代表値（A4、B4、Letter）を代表値として選択して組合せを取るという方法であった。なお、通常の直交表では3水準がMAXとなっているため、代表値は3つが限度であった。

ところが、近年、ソフトウェアの複雑化が増し、一つの機能の中に多くのロジックが含まれているため代表値は、2～3では済まなくなって来ている。HAYST法では2水準系の大きな直交表L128を用い、これを変形させることにより1因子あたり8水準まで対応している。また、代表値を選ぶという方法ではなく、ドメイン単位で割り付けを実施し、その後、ドメインごとに同値クラスの値を割り振っている。これによりド

メイン単位の直交性を確保しながらほぼ 100%の水準がテストマトリックスに現れるようになった。

2.2. 禁則回避処理

通常の実験計画法で扱われる因子は、環境（温度、湿度等）や素材の質（硬度、粘度）の組合せを対象としている。この場合「組合せることができない」ことについては、考慮の必要が無い場合が多い。

しかし、ソフトウェアの場合は、同時に選択できない機能（OHP シートに両面印刷は意味が無い等）が多数存在している。また、A を選択しないと B が選択できないといった入れ子状態の制約（カラーモードを選んで初めて色調整ができる等）も存在する。HAYST 法ではこれらの制約のことを禁則と呼んでいる。

禁則処理は複雑に絡み合っている。図２はプリンタドライバの禁則関係の一部を取り出したものである。

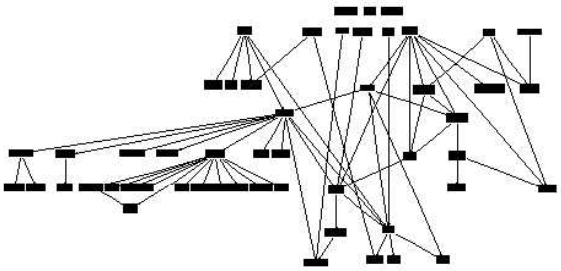

図 2　禁則関係のグラフ

禁則関係は図２のように、ある因子は複数の因子と禁則関係を持ち、それらが階層になってつながるグラフに整理できる。さらに禁則には方向性を持つものが存在する、つまり、A を設定してから B を選択する場合とその逆で B を設定してから A を選択する場合では、内部ロジックが異なっているケースがある。HAYST 法では正常系のみの（禁則にかからない）組合せを生成することを目的としているため、禁則の方向については単一方向（上から下）のみを考慮している。

異常系（禁則自体のテスト）については、方向性を考慮する必要があるが、プリンタドライバ程度のソフトウェアで方向を考慮すると 20 万を超える禁則があることが分かっている。弊社の場合、これら異常系については、ソフトウェアに入力されないように、GUI で入力を制限することが基本となっている。このGUI のエラー表示確認テスト自動化する場合に従来の記録再生型のツールを適用すると 20 万件のテストを一度は実行する必要があるため実用にはならない。そこで、因子、水準を入力するだけで、その他の操作、つまりテスト対象の因子、水準を選択するまでに必要な操作をダイナミックにツールが補完する自動実行ツール[10]を用いて確認していくことが有効と考えている。

これら複雑な禁則に対し、直交表にその組合せが出現しないように、HAYST 法では、禁則関係を表にまとめた後に、因子融合処理、多層化処理、可変因子処理を用いて禁則を回避している。

因子融合処理は、図３にあるような排他関係にある禁則に適用する。

表１　融合後の因子

用紙種類・両面
普通紙・しない
普通紙・長辺とじ
普通紙・短辺とじ
OHP・しない
厚紙・しない
ラベル紙・しない

図 3　排他関係

因子融合処理では、これらの因子を単独に直交表の列に割り付けるのではなく、表１のように融合してから割り付ける。

多層化処理は、図４にあるような、何かを選択して初めて機能が選択できる入れ子状態の禁則に対して適用する。

図 4　多層化

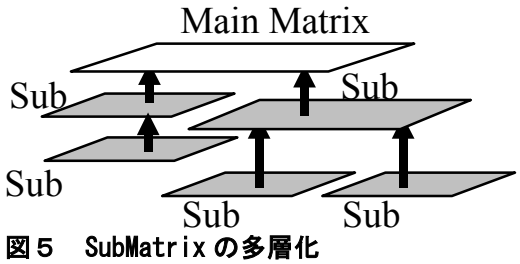

図５　SubMatrix の多層化

多層化では、入れ子になった因子を別の直交表に割り付けてからより上位の直行表に連結することで禁則を回避している。

可変因子処理は、その他の禁則パターンの回避方法

であるが、実験計画法の変身法を基本としている。

2.3. 組合せ網羅率をテスト品質指標として採用

ソフトウェアテストの終了基準は、バグ曲線の飽和状況が一般的に使用されてきた。しかし、２機能間の組合せを利用したテストによる不具合検出の重要性が D. M. Cohen [8] らによって報告されており、当社の分析結果でも同様の結果を得ている。そこで、ソフトウェアテストの品質指標として、２水準間の組合せ網羅率を使用することにした。

組合せ網羅率 [%]

$$= \frac{\text{マトリクスに現れる2水準間の組合せ数}}{\text{2因子間の全組合せ数} - \text{2因子間の禁則組合せ数}} \times 100$$

この数値が高いほど、テストの完全性は高い。従来の、勘と経験でテストマトリクスを作成した場合は30％～40％程度であることがわかっている。一方、HAYST 法の場合、70％～90％程度の網羅率が実現できている。

2.4. バグの即時原因解析

直交表での不具合解析は、通常全テスト終了後に実施される[6]。小さな直交表を使う場合はそれでもよいが、大きな直交表では終了までに数日を要するため、事後解析では終了までにソフトウェア開発者の待機が発生する。そこで、HAYST法では、同じ組合せが複数回出現するという直交表の性質を利用して、テストと解析の同時進行を行っている。不具合が発生した場合、①不具合項目が発生したテスト項目について全因子組合せを洗い出す。②実施済みのテスト項目を分析し、前段とのANDを取る。③原因組合せを特定するために、残りのテスト項目の中で、どのテスト項目を優先的に実施するかを表示する。

以上により、残りのテスト項目の消化と、分析作業を同時進行にし、分析のためにテスト進行を止める無駄と、開発者の待機の無駄を省いた。

3. HAYST 法の適用結果

プリンタドライバの機能組合せテストに、従来法と並行して、HAYST 法を試行的に適用した。従来法では 2 水準間の組合せ網羅率は 30％であったが、HAYST 法でテストマトリクスを作成した結果、組合せ網羅率は 2.5 倍向上し、不具合抽出数が 4.5 倍に増加した（図６）。

この結果をもとに、自動割り付けツールの開発と、他製品への展開を行った。

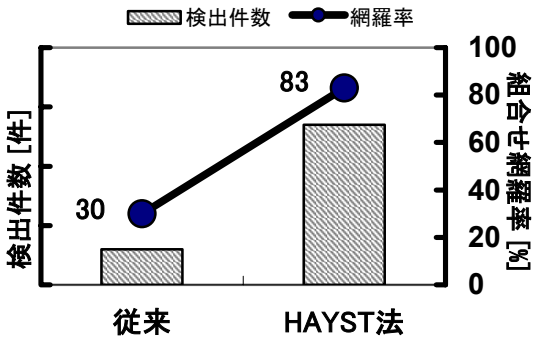

図６　網羅率とバグ検出力

ツールの開発では、誰でも使え、且つ、すぐにテストマトリクスの品質を確認できることを目指した。

マトリクスを作成するための入力は、ソフトウェアの仕様に基づいた、“因子水準”および“禁則”と、直交表の制約に基づいた、“線点図割付け情報”のみである。品質を確認するための機能として、定量的データを出力するための組合せ網羅率計算機能と、２機能間の総当り表を作成する機能を実装した。

ツール化を実現することにより、従来の経験型（蓄積された情報に基づいたスクリプト作成）と比較し、テストの設計、すなわち、どの因子水準をテストに盛り込むのかを検討することに時間がかけられるようになり、かつ全体としては工数を削減することが出来た。

また、テスト設計者の経験に依存していたテストスクリプトの質のばらつきを抑えられるようになった。

4. まとめ

L128 までの２水準系直交表を用い、多因子多水準系に対応した。直交表に割り付けるうえで、禁則情報から禁則構造を解析し、禁則を回避できるようにした。

これらの生成プロセスを自動化し、誰でも同一品質のテストマトリクスを生成できるようにした。この結果、２水準間の組合せ網羅率が大幅に向上し、バグ抽出が向上した。ここから、組合せ網羅率の重要性も示すことができた。

5. 今後の取り組み

HAYST 法自体の拡張については、16 水準×16 水準の全ペア確認が可能な L256 への対応を実施している。

また、バグは全体に散らばっているのではなく、局所的にまとまって存在していることが多いことから、部分的に網羅率を上げることができる機能をツールに組み込もうとしている。この機能追加により 1.1 で述べた直交表実験の欠点（＝多次元への対応）が補えるのではないかと考えている。

HAYST 法は、ソフトウェアの正常系の機能組合せテストに効果があることがわかった。しかし、実際にソフトウェアを使用するユーザは、正しい入力をするとは限らない。ここで重要となるのは、正しい入力をさせるように導く操作性である。筆者らはソフトウェアの操作性についての評価を定量的に実施する方法をもっていない。

また、MDA などにより設計がシステム化していることから、それら UML などで書かれた設計書（シーケンスダイアグラム等）から情報を取り出し、HAYST 法と組合せ、設計段階で並行してテスト設計を実現できるようにしたいと考えている。

文　献

[1] 田口玄一, 実験計画法　上下, 丸善, 1957-1958.

[2] 保田勝通, ソフトウェア品質保証の考え方と実際, 日科技連, 1995.

[3] 日本能率協会, 富士通の高信頼性運動, 日本能率協会, 1985.

[4] 久保宏志, 富士通におけるソフトウェア品質保証の実際, 1989.

[5] 佐藤忍、下川浩樹, 実験計画法を用いたソフトウェアのテスト項目設定法, 日科技連第 4 回 SPC シンポジウム発表法文集, pp.1-8, 日科技連, 1984.

[6] 田口玄一, デバッグのためのテスト計画, 標準化と品質管理, Vol.52, No.7, pp84-90, 1999.

[7] 高田圭、他, 直交表を使ったソフトウェアのバグ発見の効率化, 品質工学, Vol.8, No.1, pp60-64, 2000.

[8] D. M. Cohen, An Approach to Testing Based on Combinatorial Design, IEEE, Vol.23, No.7, 1997.

[9] G. J. Myers, The Art of Software Testing, John Wiley & Sons, New-York, 1979, ソフトウェア・テストの技法, 近代科学者.

[10] 小坂史、山口聡之, ダイナミックに操作を補完するテスト自動化について, ソフトウェアテストシンポジウム 2003 予稿集, pp21-28, 2003.